μLab

User’s Manual for μLab 3

revision 1.0

the innovation will play forever

SIMNUMERICA

Contents

Chapter 1

Getting Started

This chapter describes the preliminary steps you should perform to start using $\mu Lab^{®}$.
$\mu Lab^{®}$ is meant to be executed as a cloud application, providing the following advantages for its end-users:

- since all the heavy work is being performed in the cloud server, the only system requirement we demand from the developer perspective is a reasonable internet connection; actually, a common standard connection is sufficient to smoothly interact with $\mu Lab^{®}$;
- because of this, you don't need to update any of your computer hardware;
- furthermore, because all of your projects run in a cloud server, you can access and work on them from any locations and devices;
- multiple access are allowed, to give to a distributed teamwork the possibility to interact within the same simulation.

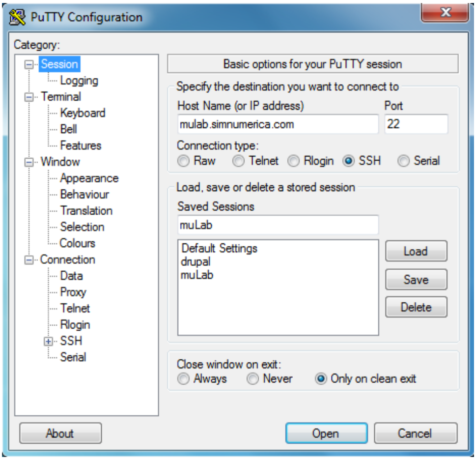
Let us describe all the steps needed to get $\mu Lab^{®}$ up and running, as well to provide safety and security to all the data transitioning from your computer/tablet to the server and viceversa. The current version of this manual assumes by default that you are using the Windows operating system on your client computer, but the same operations can be mad also with other other systems (Linux, Mac), as explained in the *Linux/Mac users* subsection of each section.

1.1 Register on the web site and activate your muLab account

TO DO: UNDER CONSTRUCTION ...

1.2 Installing an ssh client (putty) and setting the remote connection

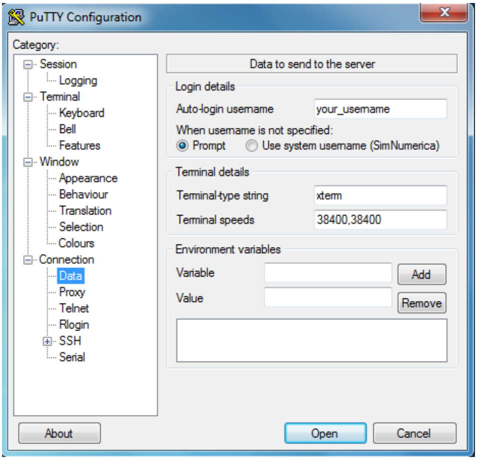
As mentioned before, data security and safety are of primary concern. Therefore, all communications between your local machine and the server hosting $\mu Lab^{\circledR}$ should take place through an encrypted SSH connection.
To set up the connection, you first need to download and configure *PuTTY*, freely available here.
Place a shortcut to it in a convenient place, such as in the desktop or taskbar, then launch it. The first option to configure inside PuTTY is the hostname to connect. From the *Category* sidebar, select *Session* subcategory; then, locate the label *"Hostname (or IP address)"* and, in the text field immediately below it, enter the hostname **mulab.simnumerica.com**
.
A screenshot of this step is shown in Figure 1.1.



Specify the destination you want to connect to
Host Name (or IP address): mulab.simnumerica.com
Port: 22
Connection type:
○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial
Load, save or delete a stored session
Saved Sessions: muLab
Default Settings
drupal
muLab
Close window on exit:
○ Always ○ Never ● Only on clean exit

Figure 1.1: Hostname configuration

Please ignore for now the *Session* entries: we will come back later. Now, you need to supply your login credentials to PuTTY, so that it can forward them to the cloud server. To perform this simple operation, just head to the *Connection* -> *Data* subcategory, and locate the *Auto-login username* label. On its right, instead of `your_username` shown in Figure 1.2 as example, insert the username you registered in the *µLab*® registration page.



Data to send to the server

Login details

Auto-login username: your_username

When username is not specified:
● Prompt ○ Use system username (SimNumerica)

Terminal details

Terminal-type string: xterm

Terminal speeds: 38400,38400

Environment variables

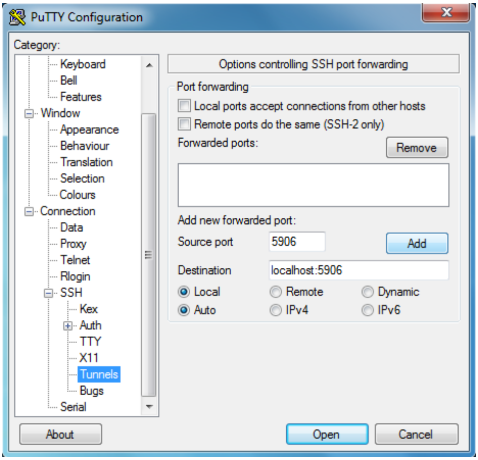
Variable:

Value:

Figure 1.2: Username configuration

At this point, we are almost done with PuTTY configuration: it knows to which server he has to connect, and which username present to it. What it is still missing, is to define how to convey the remote desktop session in the SSH secure channel. In other words, we have to define now a tunnel connection.
To do so, head to the *Connection* -> *Data* -> *SSH* -> *Tunnel* subcategory, and locate the *Source port* and *Destination* input boxes. Recalling the `port_number` you got when you received the activation email response (cfr. sec. 1.1), insert the `port_number` in the *Source port* and, as *Des-*

tination, insert `localhost:port_number`. See example in Figure 1.3, with `port_number` equal to 5906.
Click *Add* to create a tunnelled connection, and you are done. Now, to



□ Local ports accept connections from other hosts
□ Remote ports do the same (SSH-2 only)

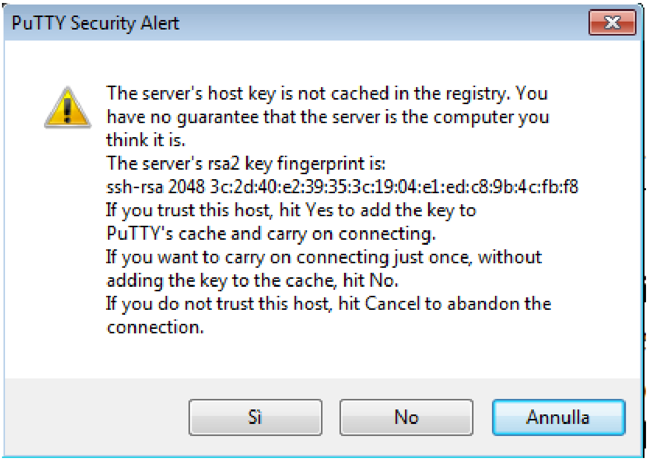
Source port: 5906
Destination: localhost:5906

● Local ○ Remote ○ Dynamic
● Auto ○ IPv4 ○ IPv6

Figure 1.3: Tunnel configuration

save your time for every future connection, you can store this session profile and load it when you need it. To do so, head back to *Session* category, below the *Saved Sessions* label type a sensible name for that connection (in Figure 1.1 we saved it as `muLab`), and then click *Save*.

Now you can test your connection: double click on your earlier saved session and a popup will show up, asking to trust our server. You just need to answer *Yes* once: in subsequent connection attempts, that popup will not show again.



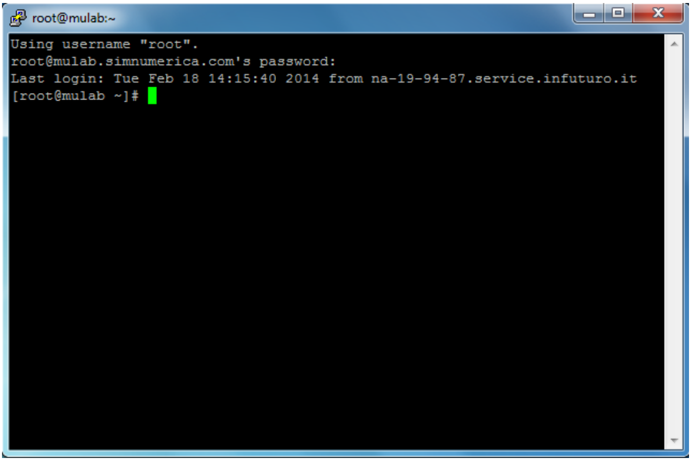
PuTTY Security Alert

The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 3c:2d:40:e2:39:35:3c:19:04:e1:ed:c8:9b:4c:fb:f8
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

Figure 1.4: Server trust request

Once you trusted our server RSA key, the command prompt will show up. Enter your password, the one you choose when you registered in the $\mu Lab^{®}$ registration page, and you will see a command line prompt pointing at your home directory, located inside the cloud server. See Figure 1.5 as reference.

The command line is not the direct interface you will use in your working session. But, it activates the secure, tunnelled connection that VNC will use to display the remote desktop, which is the interface you will use to interact with $\mu Lab^{®}$ (see sec. 1.3).



```
Using username "root".
root@mulab.simnumerica.com's password:
Last login: Tue Feb 18 14:15:40 2014 from na-19-94-87.service.infuturo.it
[root@mulab ~]#
```

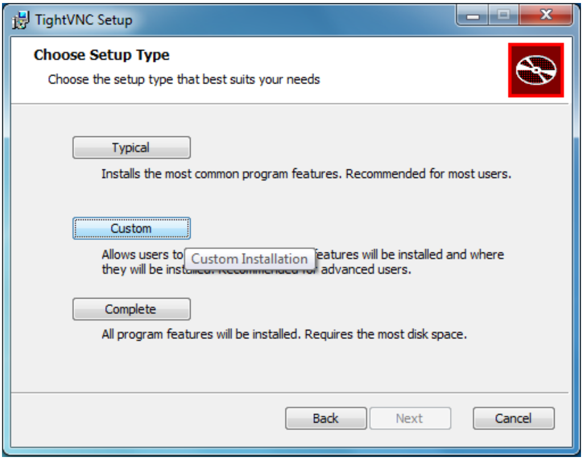
Figure 1.5: Successful connection to the remote server

1.2.1 Linux/Mac users

TO DO: UNDER CONSTRUCTION ...

1.3 Install VNC viewer and connect to your remote desktop session

Now you need to setup the application that will allow you to show the remote desktop and work on it as if you were sitting in front of the cloud server.
For this operation, you have to install *TightVNC*, freely available here. Download and install it as you would do with any other application, just remember not to install *TightVNC Server* because you only need its client application to connet to our server. Therefore, choose the *Custom* installation mode as shown in the figure above, and then click on the *TightVNC Server* entry and choose the option *Entire feature will be unavailable*.



Choose Setup Type

Choose the setup type that best suits your needs

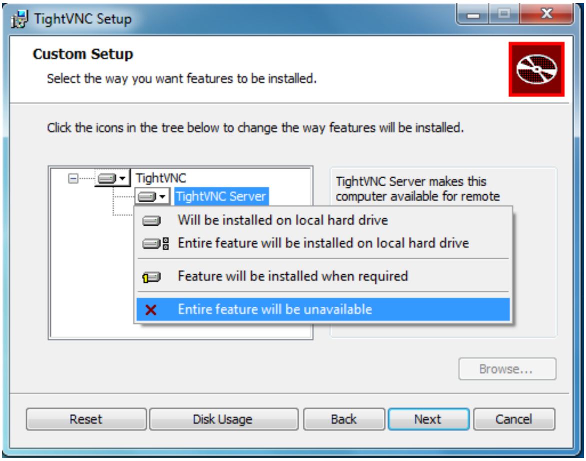
Installs the most common program features. Recommended for most users.

Allows users to eatures will be installed and where they will be inst advanced users.

All program features will be installed. Requires the most disk space.

Figure 1.6: TightVNC installation

At this point, with the PuTTY command line still active, run the *TightVNC Viewer* application: its interface is really minimal, you only have to specify the remote host and your specific port number, and then hit *Connect*



TightVNC Setup

Custom Setup

Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

TightVNC Server makes this computer available for remote

TightVNC
TightVNC Server
Will be installed on local hard drive
Entire feature will be installed on local hard drive
Feature will be installed when required
Entire feature will be unavailable
Browse...
Reset
Disk Usage
Back
Next
Cancel

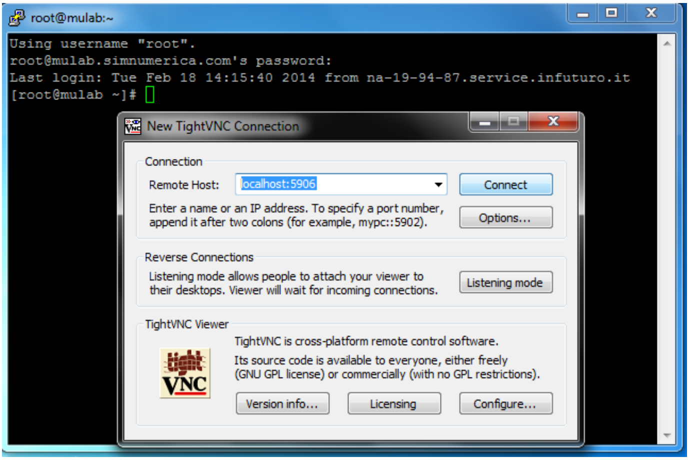
Figure 1.7: TightVNC installation

as shown in Figure 1.8 .

⚠ **Warning**: PuTTY connection **must** be running during the whole VNC session. If you closed it, either accidentally or on purpose, you will lose your remote desktop connection

If everything was setup correctly, now you will be presented a messagebox asking for your password to login into your remote desktop environment.

Enter your password, and you are done! Welcome to your remote desktop environment: you can start using *µLab*® .



```
Using username "root".
root@mulab.simnumerica.com's password:
Last login: Tue Feb 18 14:15:40 2014 from na-19-94-87.service.infuturo.it
[root@mulab ~]#
```

New TightVNC Connection

Connection

Remote Host: localhost:5906

Enter a name or an IP address. To specify a port number, append it after two colons (for example, mypc::5902).

Reverse Connections

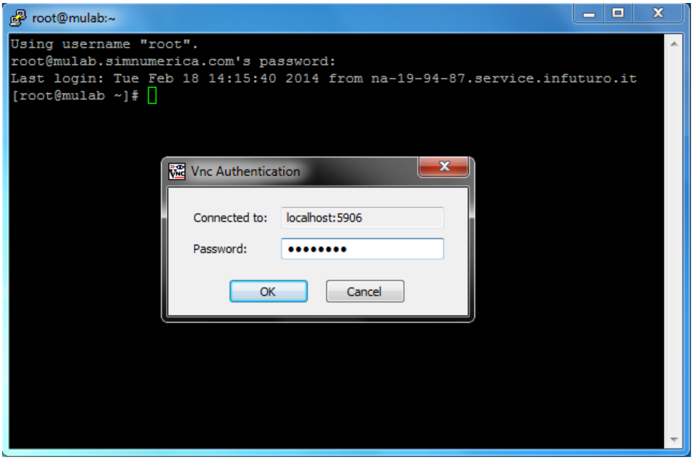
Listening mode allows people to attach your viewer to their desktops. Viewer will wait for incoming connections.

TightVNC Viewer

TightVNC is cross-platform remote control software.

Its source code is available to everyone, either freely (GNU GPL license) or commercially (with no GPL restrictions).

Figure 1.8: VNC configuration



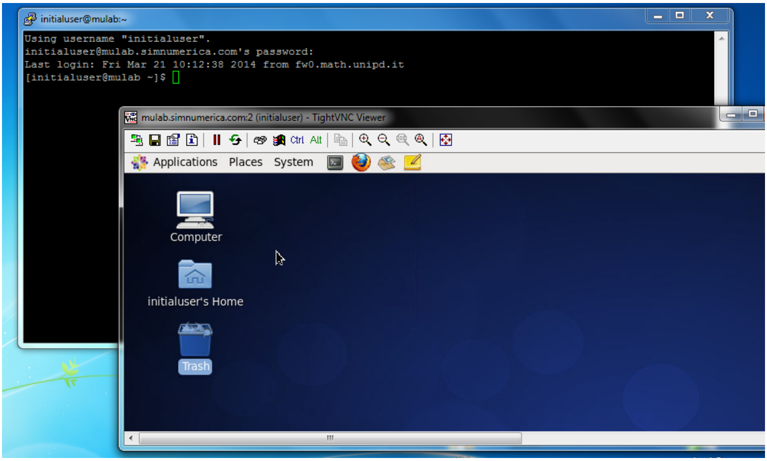
```
Using username "root".
root@mulab.simnumerica.com's password:
Last login: Tue Feb 18 14:15:40 2014 from na-19-94-87.service.infuturo.it
[root@mulab ~]#
```

Vnc Authentication

Connected to: localhost:5906

Password: ●●●●●●●●

Figure 1.9: VNC authentication



```
Using username "initialuser".
initialuser@mulab.simnumerica.com's password:
Last login: Fri Mar 21 10:12:38 2014 from fw0.math.unipd.it
[initialuser@mulab ~]$
```

Figure 1.10: VNC remote desktop

For a better view, you can expand the window to fit your screen ... the *μLab*® GUI is designed for a full-screen usage.
If you wish, you can also set your VNC client to run in fullscreen mode. In case of TightVNC, the shortcut to toggle fullscreen mode on and off is *CTRL+SHIFT+ALT+F*.

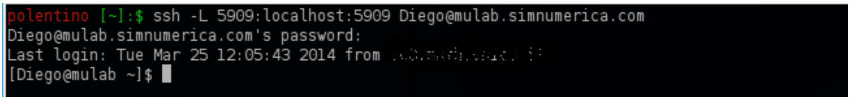
⚠ **Warning**: If you think you may forget that shortcut combination, especially after a long session development, you are encouraged to write it down. This is because you won't be able to switch to another application (e.g. to read the PDF of this manual) until you exit the fullscreen mode.

1.3.1 Linux/Mac users

To setup a successful connection to the remote desktop under Linux/Mac, you have to open your favourite terminal application and type, in one single line:

```
ssh -L <port_number>:localhost:<port_number>
<username>@mulab.simnumerica.com
```

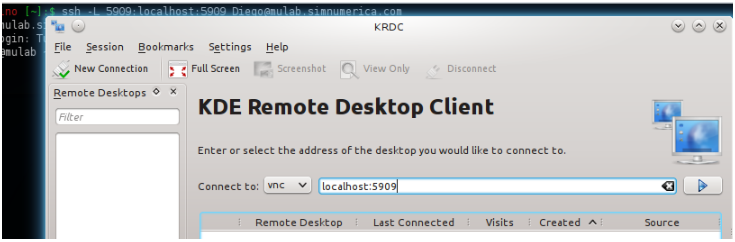
where <*port_number*> must be substituted with your actual port number (the one you received by email) , and <*username*> of course by your username in the remote server.
In the example below, we performed tunnelling of port 5909 for user *Diego*



```
polentino [~]:$ ssh -L 5909:localhost:5909 Diego@mulab.simnumerica.com
Diego@mulab.simnumerica.com's password:
Last login: Tue Mar 25 12:05:43 2014 from [redacted]
[Diego@mulab ~]$ █
```

Figure 1.11: Linux/Mac tunneling

Now, the steps to connect to your remote desktop are the same shown in Windows: just use your favourite VNC client shipped with your distribution, insert *localhost:<port_number>* as remote local host and click *Connect*: once you provide your remote VNC password, you are done.
If, for example, you are using KDE's remote desktop application *KRDC*, simply insert *localhost:<port_number>* and then click the blue arrow on the right, as shown in figure



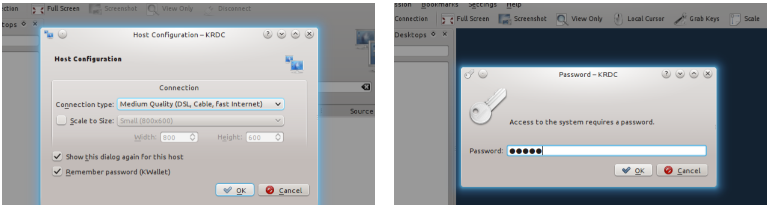
```
no [~]:$ ssh -L 5909:localhost:5909 Diego@mulab.simnumerica.com
```

KDE Remote Desktop Client

Enter or select the address of the desktop you would like to connect to.

Connect to: vnc localhost:5909

Figure 1.12: KRDC host spelification



Host Configuration – KRDC

Host Configuration

Connection

Connection type: Medium Quality (DSL, Cable, fast Internet)

☐ Scale to Size: Small (800x600)

Width: 800 Height: 600

☑ Show this dialog again for this host

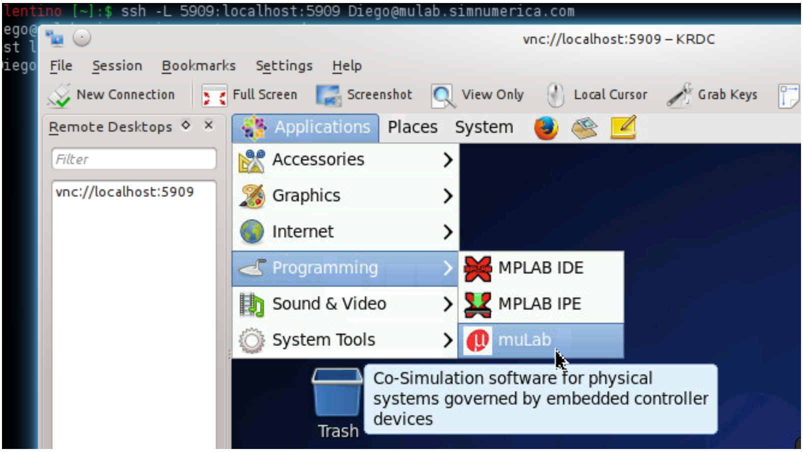
☑ Remember password (KWallet)

Password – KRDC

Access to the system requires a password.

Password: ●●●●●

Figure 1.13: KRDC config dialog, followed by the password dialog



```
lentino [~]:$ ssh -L 5909:localhost:5909 Diego@mulab.simnumerica.com
```

vnc://localhost:5909 – KRDC

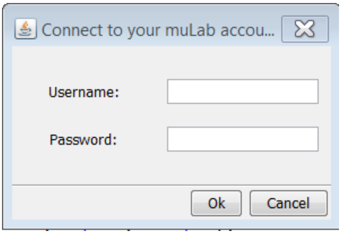
vnc://localhost:5909

Co-Simulation software for physical systems governed by embedded controller devices

Figure 1.14: KRDC with remote desktop active on user Diego

1.4 Run *μLab®*

From the *Applications* menu, located in the upper left side of the screen, select the *Programming* submenu and launch *μLab®* , by clicking the *μLab®* menu entry. After a short time it will appear the *μLab®* login interface, as shown in Figure 1.15: enter your username and password, and start using the *μLab®* environment!


Connect to your muLab accou...

Username:

Password:

Figure 1.15: *μLab®* login interface

If you want to place a shortcut in the desktop instead, just right click over *μLab®* menu entry and select *Add this launcher to desktop,* as shown in Figure 1.16.

After the program has finished loading, the main *μLab®* dashboard will be shown (see Cap. 2). You can use it to access all of the *μLab®* functions and tools; upon being started, *μLab®* loads by default the *TestBench* used in the last session, restoring graphical properties such as the icon positions on the graph, the size of the main window, ...

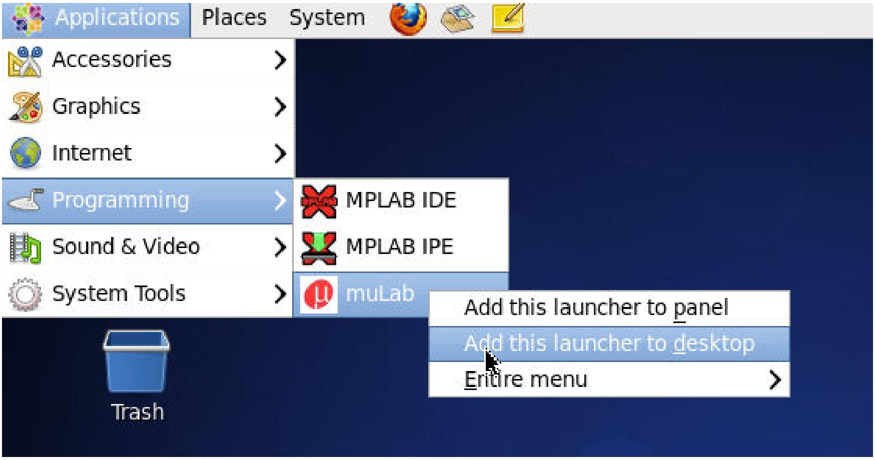


Figure 1.16: Run $\mu Lab^{®}$ in the remote desktop

⚠ **Warning**: Once you have completed your $\mu Lab^{®}$ session, **do not logout** from your login in the server. If you do so, you won't be able to reconnect with your remote desktop until the server administrator reactivate your login. Instead, just close VNC and PuTTY: once you re-connect again, you will access the same working environment left at the end of your previous session (this can be very useful: for example, if you left a simulation running, you will access to its current status and interact with it).

1.5 Copying your files into your μ*Lab*® remote account space

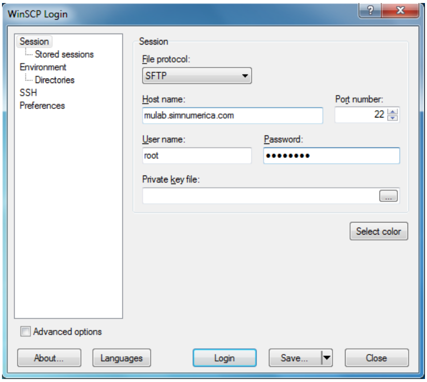
You have seen how to setup a secure connection, how to display your remote desktop session, and how to launch mulab. The natural step that follows is to start composing one or more *TestBench*, i.e. your virtual prototyping projects within μ*Lab*® : this requires also providing custom firmwares and some Python scripts (to model the physical world interacting with your controller, to perform some postprocess actions, etc.) that may have been written within another computer system.
In principle, you could develop everything from within your remote desktop session but, what your are most likely going to need, is a way to transfer *TestBench* files and simulation datafiles from and to your remote account space. Thus, you need to download and install *WinSCP* from here (free application).
Its interface is simple and intuitive, as well its setup: once you started it, click on *New* button located in the upper rigth side of the application, and the interface will change to let you enter details about the session you are going to establish, as you can see in the image below

You only need to fill the following fields:

- *Host name* with `mulab.simnumerica.com` ;
- *User name* with `your_username` ;
- *Password* with `your_password`, the same used with PuTTY (cfr. sec. 1.2);

After doing so, you can either press the *Login* button or, and we advise you to do so, save this new session and then double click it from the *Stored sessions* list.

If you did type everything correctly, after the connection phase you will see a new window similar to the one shown in Figure 1.18: on the left, you have your local directories and files; on the right, you can explore your remote account directory and subfolders. If you want to upload a file or folder from your local computer to the server, just drag and



File protocol: SFTP

Host name: mulab.simnumerica.com

Port number: 22

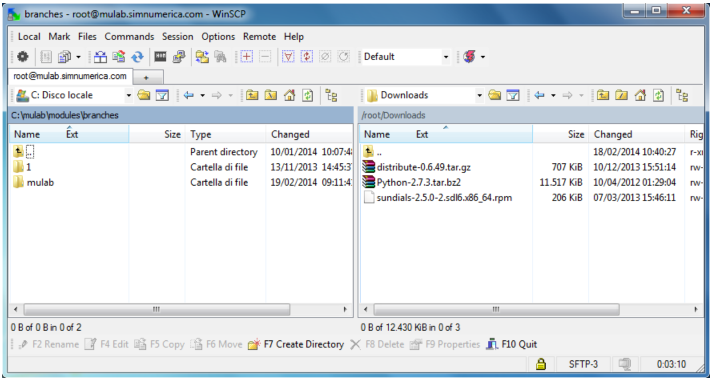
User name: root

Password: ••••••••

Private key file:

□ Advanced options

Figure 1.17: WinSCP setup page



C:\mulab\modules\branches

Name	Ext	Size	Type	Changed
..			Parent directory	10/01/2014 10:07:4
1			Cartella di file	13/11/2013 14:45:3
mulab			Cartella di file	19/02/2014 09:11:4

0 B of 0 B in 0 of 2

/root/Downloads

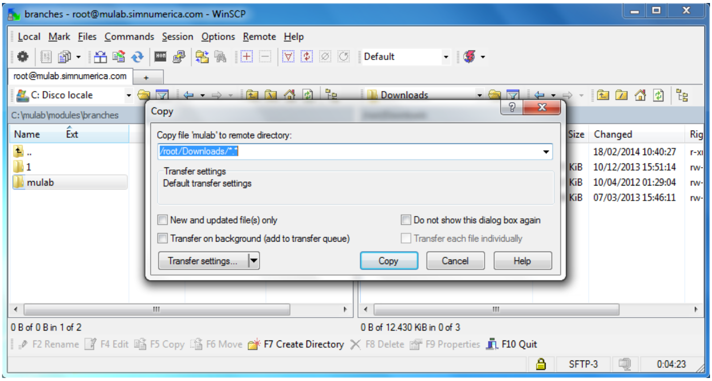
Name	Ext	Size	Changed	Rig
..			18/02/2014 10:40:27	r-x
distribute-0.6.49.tar.gz		707 KiB	10/12/2013 15:51:14	rw-
Python-2.7.3.tar.bz2		11.517 KiB	10/04/2012 01:29:04	rw-
sundials-2.5.0-2.sdl6.x86_64.rpm		206 KiB	07/03/2013 15:46:11	rw-

0 B of 12.430 KiB in 0 of 3

Figure 1.18: WinSCP connected to mulab remote account space

drop it from the left panel to the right panel. In Figure 1.19 for example, we copied the local folder *mulab* inside *C:\mulab\modules\branches* into the remote folder */root/Downloads*. Win-

SCP is smart enough to provide some transfer options to better improve the transfer rate and reduce bandwidth usage: feel free to try them and then press the *Copy* button.



branches - root@mulab.simnumerica.com - WinSCP

Copy

Copy file 'mulab' to remote directory:

/root/Downloads/*.*

Transfer settings
Default transfer settings

□ New and updated file(s) only
□ Do not show this dialog box again
□ Transfer on background (add to transfer queue)
□ Transfer each file individually

Figure 1.19: WinSCP folder upload to remote server

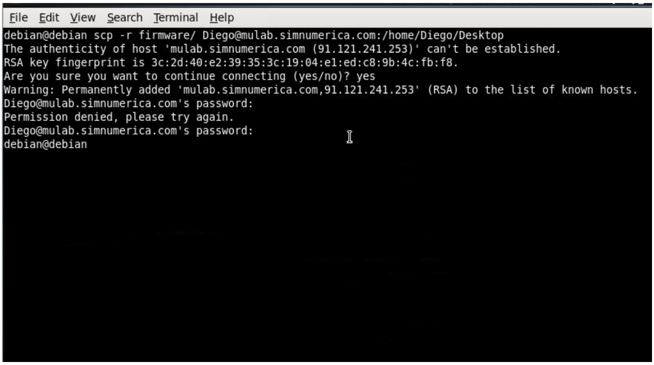
1.5.1 Linux/Mac users

If you are running any GNU/Linux distribution, you can copy any file and folder with the handy *scp* command. It takes the following parameters:

- the file or folder you need to transfer (when transferring folders, remember to add the *-r* parameter to send the folder and its content, recursively);

- the username in the cloud server, hostname, and destination folder (absolute path) where the file/folder will be placed.

An example of transfer operation of folder *firmware* (empty in that case), executed by user Diego, into his own remote Desktop folder, is show in the figure below: The first time you perform this operation, scp will ask you if you are sure to connect with the host: just reply *yes*, and the server credentials will be saved for the future connections.



```
debian@debian scp -r firmware/ Diego@mulab.simnumerica.com:/home/Diego/Desktop
The authenticity of host 'mulab.simnumerica.com (91.121.241.253)' can't be established.
RSA key fingerprint is 3c:2d:40:e2:39:35:3c:19:04:e1:ed:c8:9b:4c:fb:f8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'mulab.simnumerica.com,91.121.241.253' (RSA) to the list of known hosts.
Diego@mulab.simnumerica.com's password:
Permission denied, please try again.
Diego@mulab.simnumerica.com's password:
debian@debian
```

Figure 1.20: Folder transfer operation using *scp* command

TO DO: UNDER CONSTRUCTION ...

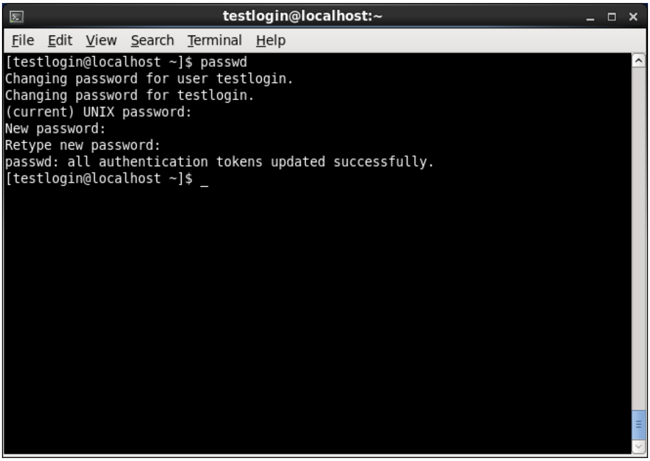
1.6 Final touches

Now that you are able to setup a secure connection to the cloud server,copy files and folders into it, and start $\mu Lab^{®}$, there are some final touches you are likely interested to perform as well.

1.6.1 Change password

For example, you may want to change your login password and your VNC password to something different than the ones you received from the administrator email. This process is really fast and easy to perform, so we highly encourage you to do so.

To change your *user login password*, open a new terminal and type the `passwd` command, as shown in Figure 1.21.

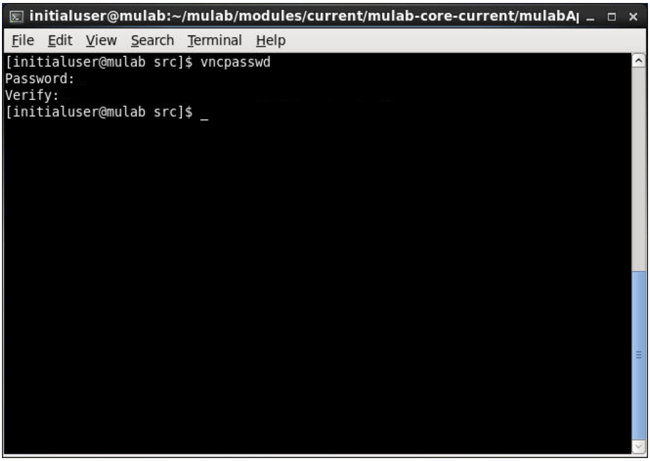


```
[testlogin@localhost ~]$ passwd
Changing password for user testlogin.
Changing password for testlogin.
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[testlogin@localhost ~]$ _
```

Figure 1.21: Login password change

The prompt will ask for your current password and then it will ask

you to enter your new password twice. If everything went good, the prompt will inform you that your authentication tokens were updated successfully.
An almost identical process applies if you change VNC password, as you can see in Figure 1.22.


initialuser@mulab:~/mulab/modules/current/mulab-core-current/mulabA|
File Edit View Search Terminal Help

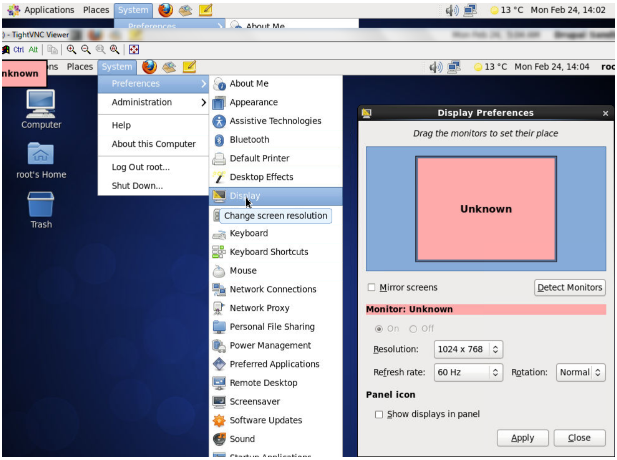
```
[initialuser@mulab src]$ vncpasswd
Password:
Verify:
[initialuser@mulab src]$ _
```

Figure 1.22: VNC password change

In fact, now you have to type the command `vncpasswd` and, as opposed to `passwd`, it will not ask for your current vnc password but it will simply ask for a new one to set.

1.6.2 Change desktop resolution

Another aspect you most likely want to change is the desktop resolution. It is a really straightforward operation if you already used Gnome desktop environment. If you never did, just follow the screenshot in Figure 1.23.



□ Mirror screens

Monitor: Unknown

● On ○ Off

Resolution: 1024 x 768

Refresh rate: 60 Hz Rotation: Normal

□ Show displays in panel

Figure 1.23: Change desktop resolution

1.7 *MPLAB-X*® issues

There are a few issues that an *MPLAB-X*® user must keep in mind, when doing co-simulations with *μLab*® .

1.7.1 *MPLAB-X IDE*® project setup

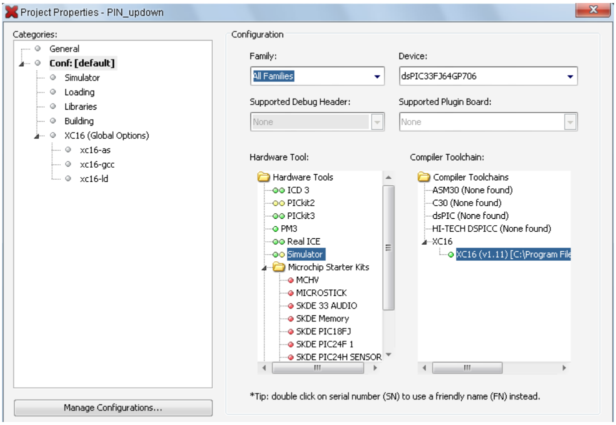
In order to perform a successful co-simulation session, you need to properly set up your *MPLAB-X IDE*® project as we will immediately show.

First, your project *must* be compiled with the new toolchains provided by Microchip along with *MPLAB-X IDE*® [1] that is, XC8/16/32 compilers.

> ⚠ **Warning**: if you import an older project that has been developed with MPLAB IDE v8 or previous versions, remember to change the compiler from mpasm or C18/30 one of its new counterpart, otherwise the plugin will misbehave and the co-simulation will not take place.

In addition to the compiler, you also need to choose the right *Hardware Tool* for the project you want to test. In fact, *μLab*® needs to query *MPLAB-X IDE*® internal simulator about the register and pinout states; therefore, you have to choose the *Simulator* option as current *Hardware Tool*. You can find both compiler and hardware tool options by right clicking the corresponding project listed in the *Project* panel. An example dialog is shown in figure 1.24.
The last detail you need to take care of is setting the right frequency for the simulator.

[1] See `http://www.microchip.com/pagehandler/en-us/family/mplabx` , *Compilers* section



Project Properties - PIN_updown

Categories:
- General
- Conf: [default]
  - Simulator
  - Loading
  - Libraries
  - Building
  - XC16 (Global Options)
    - xc16-as
    - xc16-gcc
    - xc16-ld

Manage Configurations...

Configuration

Family: All Families

Device: dsPIC33FJ64GP706

Supported Debug Header: None

Supported Plugin Board: None

Hardware Tool:
- Hardware Tools
  - ICD 3
  - PICkit2
  - PICkit3
  - PM3
  - Real ICE
  - Simulator
  - Microchip Starter Kits
    - MCHV
    - MICROSTICK
    - SKDE 33 AUDIO
    - SKDE Memory
    - SKDE PIC18FJ
    - SKDE PIC24F 1
    - SKDE PIC24H SENSOR

Compiler Toolchain:
- Compiler Toolchains
  - ASM30 (None found)
  - C30 (None found)
  - dsPIC (None found)
  - HI-TECH DSPICC (None found)
  - XC16
    - XC16 (v1.11) [C:\Program File

*Tip: double click on serial number (SN) to use a friendly name (FN) instead.

Figure 1.24: Project options

⚠ **Warning**: setting the clock frequency in *µLab*® will not have any effect on *MPLAB-X*® simulator, so please remember to do it from within *MPLAB-X IDE*® .

1.7.2 Error: Cannot start communication server

This error arises for the following three reasons:

- the plugin was not installed, or it was removed because you wanted to upgrade it, but then you forgot to install the newer version;

  Solution ⇒ contact the administrator
- there is an instance of *MPLAB-X IDE*® already running, and *µLab*® cannot connect to it because it has to start *MPLAB-X IDE*® on its own

  Solution ⇒ save everything you were working in *MPLAB-X IDE*® and close it: *µLab*® will start it and take care of its startup and communication setup

1.7.3 Warning: muLab-MPLABX plugin detected that[...]

When you see a warning message which tells you *“muLab-MPLABX plugin detected that the frequency in the simulator is different from the one set by the Clock Frequency Adapter”*, it means that you are trying to run a simulation but the frequency you set on OSC1/OSC2 is different from the one used from *MPLAB-X IDE*® internal simulation and, therefore, you will get erroneus results. To solve this issue, update the correct oscillator frequency value in the *MPLAB-X*® project, or change the frequency parameter of the AdapterClockFrequency attached to OSC1/OSC2 (see sec. 3.3.2.1).

Chapter 2

Interface Overview

The main $\mu Lab^{\circledR}$ dashboard is composed of 4 sections:

- the Menu Bar;
- the Toolbar;
- the Main Pane;
- the Status Bar;

and includes a WYSIWYG text editor for writing notes (and examining test reports, see 2.18) during your $\mu Lab^{\circledR}$ work session.

This chapter explains how to navigate the dashboard and access all of the main $\mu Lab^{\circledR}$ functionalities; you can find specific details in the corresponding sections.

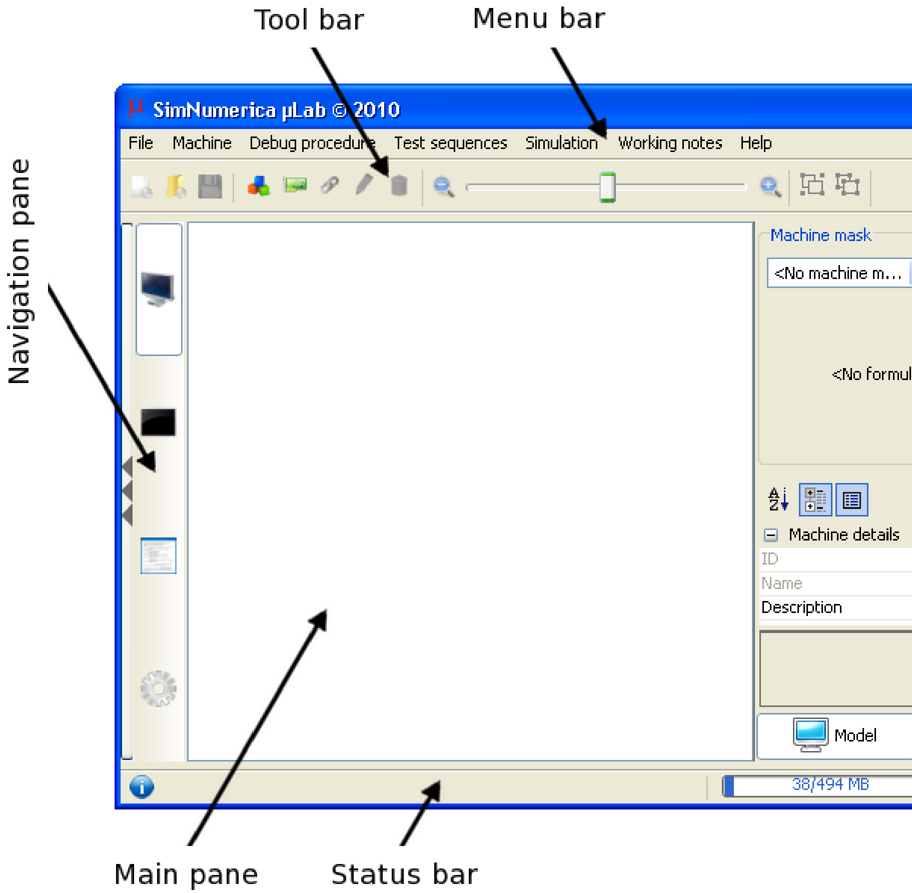


Figure 2.1: The main $\mu Lab^{®}$ dashboard

2.1 The menu bar

The menu bar contains several menus used to manage the contents of your machine and your tests.

2.1.1 File menu

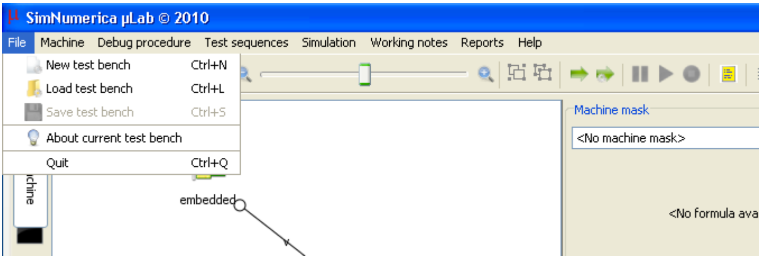


Figure 2.2: The `File` menu

The File menu (figure 2.2) is used to manage your test bench files.

New test bench: create a new, empty test bench. See also 3.1.1 for a detailed walkthrough.

Load test bench: load an existing test bench from a test bench file. See also 3.1.2 for a detailed walkthrough.

Save test bench: save the current test bench to file. This is only active if the test bench has been modified after the last save.

Rename test bench: rename the current test bench. Renaming requires the test bench to be saved (you will be asked for confirm).

About current test bench: show a brief description of the test bench you are currently working on.

Quit: terminate all of the active simulations and debug procedures and exit *μLab*® .

2.1.2 Machine menu

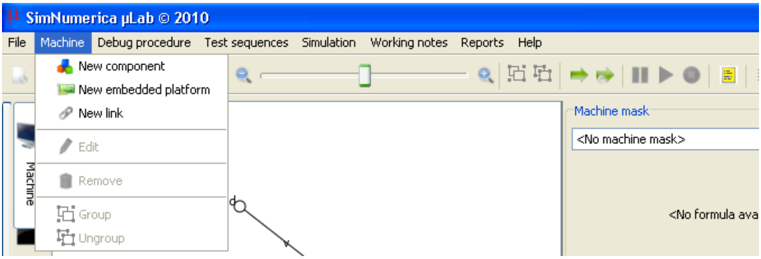


Figure 2.3: The `Machine` menu

The Machine menu (figure 2.3) lets the user define the hardware setup of the machine, by creating the components themselves and their interconnections, that model their interactions. The resulting machine can be viewed by selecting the Machine icon from the navigation bar.

New component: create a new component and add it to your machine. See also 3.2.1 for detailed instructions about creating components.

New embedded platform: create a new embedded platform and add it to your machine. See also 3.3.1 for detailed instructions about creating embedded platforms.

New link: setup a connection between two existing items. See also 3.4 for detailed instructions about links.

Edit: edit the properties of the selected item.

Remove: remove the selected item from your machine.

Group: create a group containing the selected items. See also 3.5 for more instructions about groups.

Ungroup: remove the selected group relation between items, without removing the items from the machine. See also 3.5 for more instructions about groups.

2.1.3 Debug procedures menu

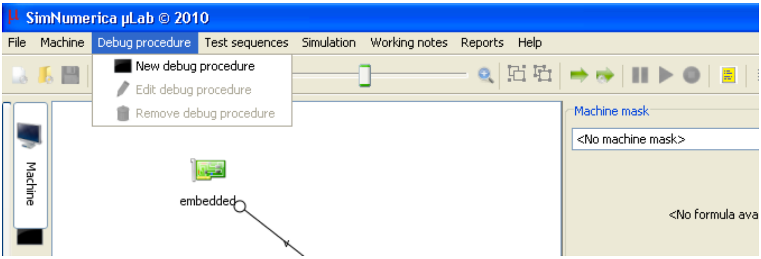


Figure 2.4: The `Debug Procedures` menu

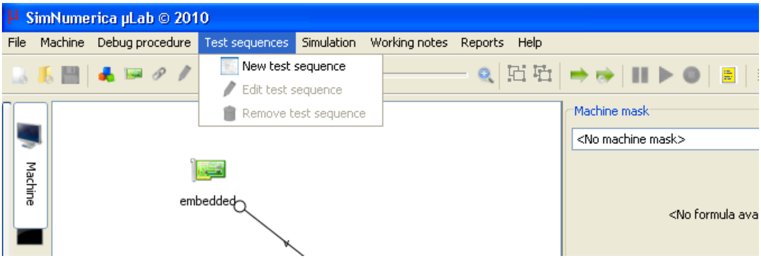
The debug procedures menu (figure 2.4) lets you create, edit and remove debug procedures. For more info about debug procedures, see Chapter 5.

New debug procedure: start the creation of a new debug procedure. See also 5.1.

Edit debug procedure: modify the properties of an existing debug procedure. See also 5.4 for editing individual debug instructions.

Remove debug procedure: delete the selected debug procedure. See also 5.3.

2.1.4 Test sequences menu



Machine mask
<No machine mask>

Figure 2.5: The `Test Sequences` menu

The test sequences menu (figure 2.5) lets you create, edit and remove test sequences. For more info about test sequences, see Chapter 6.

New test sequence: start the creation of a new test sequence. See also 6.1.

Edit test sequence: modify the properties of an existing test sequence. See also 6.5 for editing individual tests.

Remove test sequence: delete the selected test sequence. See also 6.3.

2.1.5 Simulation menu

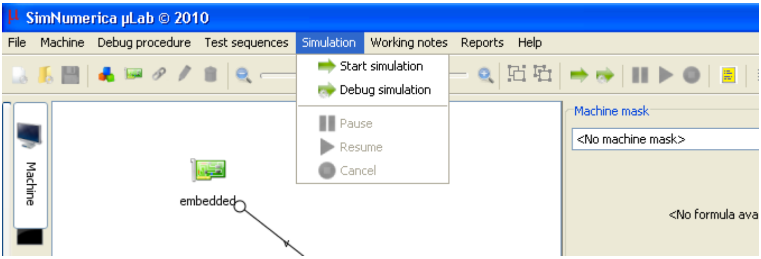


Figure 2.6: The `Simulation` menu

The Simulation menu (figure 2.6) lets you start numerical simulations on your machine, control the flow of their execution, and access the simulation docs. For more info about simulations, see also Chapter 7.

Start simulation: start the Simulation launch wizard. See also 7.1 for detailed instructions about launching simulations and test sequences.

Pause: pause the current simulation; inactive if no simulations are running. See also 7.2.1.

Resume: resume the current simulation; inactive if no simulations are active and paused. See also 7.2.2.

Cancel: abort the current simulation; inactive if no simulations are active. See also 7.2.3.

Docs: open the workbench notes and test reports managers.

2.1.6 Help menu

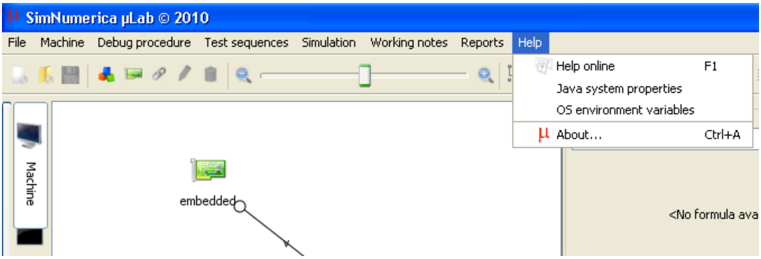


Figure 2.7: The `Help` menu

The Help menu (figure 2.7) lets you access various assistance tools.

Online Help: open the integrated *µLab*® Online Help. TO DO: SEE **??** FOR MORE INFO AND SUGGESTIONS ON HOW TO USE IT EFFECTIVELY.

Java system properties: open a viewer that lists the Java properties of your system. Cannot be used to modify their values.

OS environment variables: open a viewer that lists the environment variables of your system. Cannot be used to modify their values.

About: show an informative popup about *µLab*® .

2.2 The Tool bar

The Tool bar provides quick shortcuts for the most commonly used functions of the Menu bar. It changes with the view selected in the Navigation Bar, as described in the following sections.

2.3 The Tool bar for the Machine View

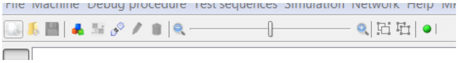


Figure 2.8: The Tool bar

Test bench buttons

- new test bench;
- load test bench.

Machine buttons

- new component;
- new embedded platform;
- new link;
- edit item;
- remove item.

Zoom bar
You can use this scroll bar to select the zoom level for the graph. Move the slider to the left to zoom out, and move it to the right to zoom in.

2.4 The Tool bar for the Simulation View

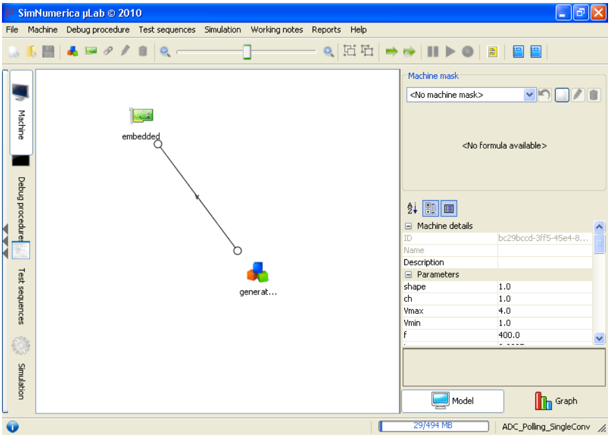
Simulation buttons

- start simulation;
- debug simulation;
- pause current simulation;
- resume current simulation;
- abort current simulation;
- open simulation docs.

2.5 The main pane

Use the Navigation Bar to choose between the Machine view, the Project Dictionaries view, the Debug Procedures view, the Test Sequences view, and the Simulation view for the main pane.

2.5.1 Machine view



Machine details

ID	bc29bccd-3ff5-45e4-8...
Name	
Description	

Parameters

shape	1.0
ch	1.0
Vmax	4.0
Vmin	1.0
f	400.0

Figure 2.9: The Machine view

The Machine view is used to show and edit the machine you're currently working on.

This view is split in two main panes: the left one is used to work on the machine structure, while the right one can be used to set up masks, view and edit variables and parameters of the model, and modify the look of the graph.

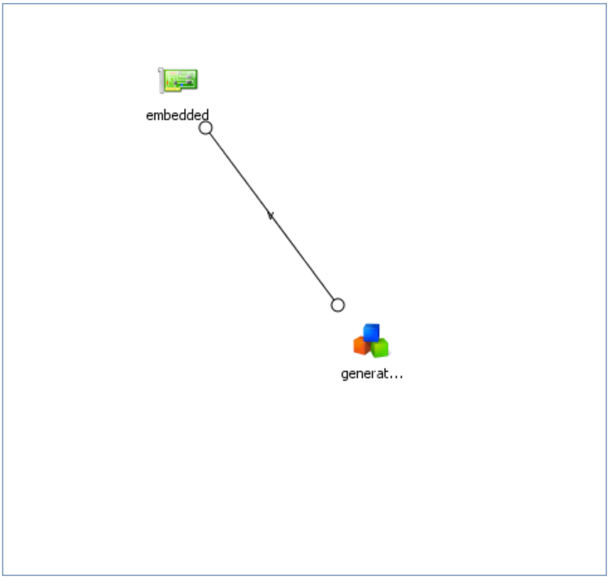
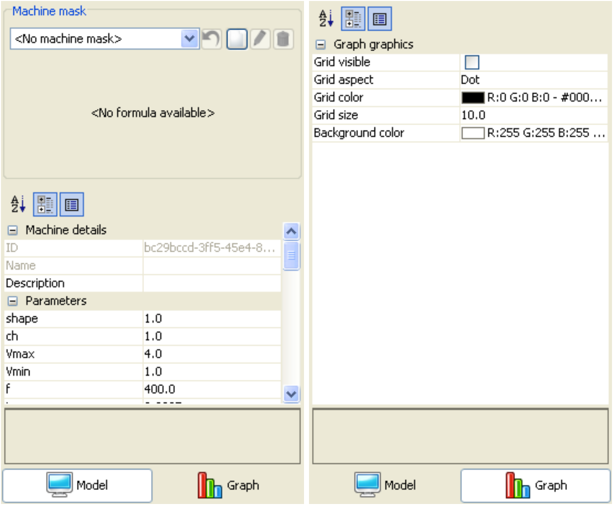


Figure 2.10: Machine view, the Graph pane

Graph pane The machine itself is represented by a graph; the nodes of the graph represent the components of the machine, and the arcs of the graph represent the links of the machine.

Component and link properties can be modified through the respective edit dialogs, accessible by double clicking the object you want to edit (or by selecting it and then choosing the `Edit` action from the `Machine` menu or the toolbar).

Components and links can be deleted through the `Delete` action, which can be selected from the Machine menu, the toolbar, or the context menu (open it by right-clicking on the component). `Del` (on your keyboard) also works as a generic shortcut for deletion.



<No machine mask>

<No formula available>

Machine details

ID	bc29bccd-3ff5-45e4-8...
Name	
Description	

Parameters

shape	1.0
ch	1.0
Vmax	4.0
Vmin	1.0
f	400.0

Graph graphics

Grid visible	□
Grid aspect	Dot
Grid color	■ R:0 G:0 B:0 - #000...
Grid size	10.0
Background color	□ R:255 G:255 B:255 ...

Figure 2.11: Machine view, the Info pane

Info pane TO DO: TROVARE NOME DECENTE PER LA SEZIONE;
The right section of the machine view consists of the Info pane. It is a panel whose contents depend on the current graph selection. TO DO: DA METTERE IN FORMA DI TABELLA

- **No items selected**

  Model view: shows machine masks, machine details, parameters (all), variables (all).

  - Machine masks: lets you choose and setup machine masks;
  - Machine details: ID, name, description of the machine;
  - Parameters: a name, value pair list of all of the parameters in the machine components;

  - Variables: a name [type], initial value pair list of all of the variables in the machine components.

Graph view: shows graph properties.

  - Grid visible: toggle this to show a grid underneath the graph (disabled by default);
  - Grid aspect: the way the underlying graph grid is drawn (crosses, dots, lines);
  - Grid color: the color of the grid;
  - Grid size: the size of the grid elements;
  - Background color: the background color of the graph.

- **One component selected**

Model view: shows component details, parameters, variables, plugs and pins.

  - Component details: ID, name, description, model file, enabled (if this option is toggled off, the status of the component will not change during simulations TO DO: CHE SIGNIFICA?;
  - Parameters: a name, value pair list of the component parameters;
  - Variables: a name [type], initial value pair list of the component variables;
  - Plugs and pins: a list containing the component plugs; each plug is composed of two name [type], initial value pairs describing its pins.

Graph view: shows vertex properties.

  - ID: the component ID;
  - Font: the font used for the component name in the graph;
  - Icon: the icon used to represent the component in the graph;
  - Background color: the background color for the component square in the graph;

  - Foreground color: the foreground color for the component square in the graph;
  - Bounds: the bounds of the component square in the graph;
  - Border visible: if toggled on, shows the border of the component square in the graph.

- **One embedded platform selected**

  Model view: shows embedded platform details, plugs and pins, processors.

  - Embedded platform details: ID, name, description, enabled (if this option is toggled off, the status of the embedded platform will not change during simulations TO DO: CHE SIGNIFICA?;
  - Plugs and pins: a list containing the embedded platform plugs; each plug is composed of two name [type], initial value pairs describing its pins.
  - Processors: a list containing the embedded platform microprocessors; each processor entry shows a list containing all of its pins.

  Graph view: shows vertex properties.

  - ID: the embedded platform ID;
  - Font: the font used for the embedded platform name in the graph;
  - Icon: the icon used to represent the embedded platform in the graph;
  - Background color: the background color for the embedded platform square in the graph;
  - Foreground color: the foreground color for the embedded platform square in the graph;
  - Bounds: the bounds of the embedded platform square in the graph;

  - Border visible: if toggled on, shows the border of the embedded platform square in the graph.

- **One link selected**

  Model view: shows link details, connections, wires.

  - Link details: ID, name, description, enabled (if this option is toggled off, the status of the link will not change during simulations TO DO: CHE SIGNIFICA?;
  - Connections: two plug type, name pairs describing (source and target) plugs connected by this link;
  - Wire: a list of pin [type], initial value pair couples describing the pins (and the initial values of the variables they carry) connected by the wires of this link.

  Graph view: shows edge properties.

  - ID: the link ID;
  - Font: the font used for the link name in the graph;
  - Line style: the line style for this link (bezier, orthogonal, spline);
  - Line begin: the style for the beginning of this link line (none, circle, simple, technical, classic, diamond, line, double line);
  - Line end: the style for the ending of this link line (none, circle, simple, technical, classic, diamond, line, double line);
  - Line color: the color of this link line in the graph;
  - Line width: the width of this link line in the graph.

- **More than one item selected**

  Model view: no properties to show.

  Graph view: no properties to show.

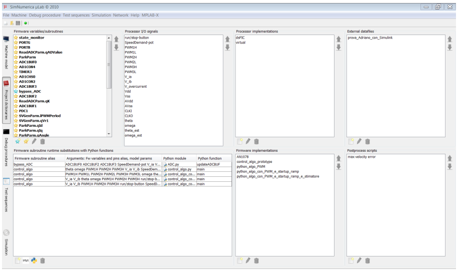
- **Groups**

  Nothing happens by clicking on groups.

2.5.2 Project Dictionaries view

The Project Dictionaries view lets you manage the dictionaries of all the relevant project items defined in the current *TestBench*. Each panel, from left to right and from top to bottom, containes the following dictionaries:

- the Dictionary of firmware variables/subroutines, cfr sec. 4.1;
- the Dictionary of processor I/O signals, cfr. sec. 4.2;
- the Dictionary of processor implementations, cfr. sec. 4.3;
- the Dictionary of external data files, cfr. sec. 4.6;
- the Dictionary of runtime firmware subroutines substitutions, cfr. sec. 4.4;
- the Dictionary of firmware implementations, cfr. sec. 4.5;
- the Dictionary of post-process scripts, cfr. sec. 4.7.



Firmware variables/subroutines

state_monitor
PORTG
PORTB
ReadADCParm.qADValue
ParkParm
ADC1BUF0
AD1CON4
TIMER3
AD1CHS0
AD1CON2
ADC1BUF3
bypass_ADC
ADC1BUF2
ReadADCParm.qK
ADC1BUF1
PDC1
SVGenParm.iPWMPeriod
SVGenParm.qVr1
ParkParm.qId
ParkParm.qIq
ParkParm.qAngle

Processor I/O signals

run/stop-button
SpeedDemand-pot
PWM1H
PWM1L
PWM2H
PWM2L
PWM3H
PWM3L
V_ia
V_ib
V_overcurrent
Vdd
Vss
AVdd
AVss
CLKI
CLKO
theta
omega
theta_est
omega_est

Processor implementations

dsPIC
virtual

External datafiles

prova_Adriano_con_Simulink

Firmware subroutine runtime substitutions with Python functions

Firmware subroutine alias	Arguments: Fw variables and pins alias, model params	Python module	Python function
bypass_ADC	ADC1BUF0 ADC1BUF2 ADC1BUF3 SpeedDemand-pot V_ia V...	ADC.py	updateADCBUF
control_algo	theta omega PWM1H PWM2H PWM3H V_ia V_ib SpeedDem...	control_algo.py	main
control_algo	PWM1H PWM1L PWM2H PWM2L PWM3H PWM3L omega the...	control_algo_co...	main
control_algo	V_ia V_ib theta omega PWM1H PWM2H PWM3H run/stop-b...	control_algo_co...	main
control_algo	V_ia V_ib PWM1H PWM2H PWM3H run/stop-button SpeedD...	control_algo_co...	main

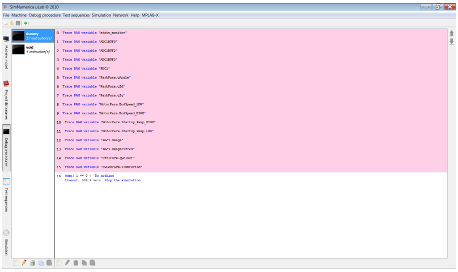
Firmware implementations

AN1078
control_algo_prototype
python_algo_PWM
python_algo_con_PWM_e_startup_ramp
python_algo_con_PWM_e_startup_ramp_e_stimatore

Postprocess scripts

max velocity error

Figure 2.12: The Project Dictionaries view



dummy
17 instruction(s)

void
4 instruction(s)

```
0  Trace RAM variable "state_monitor"
1  Trace RAM variable "ADC1BUF0"
2  Trace RAM variable "ADC1BUF2"
3  Trace RAM variable "ADC1BUF3"
4  Trace RAM variable "PDC1"
5  Trace RAM variable "ParkParm.qAngle"
6  Trace RAM variable "ParkParm.qId"
7  Trace RAM variable "ParkParm.qIq"
8  Trace RAM variable "MotorParm.EndSpeed_LOW"
9  Trace RAM variable "MotorParm.EndSpeed_HIGH"
10  Trace RAM variable "MotorParm.Startup_Ramp_HIGH"
11  Trace RAM variable "MotorParm.Startup_Ramp_LOW"
12  Trace RAM variable "smc1.Omega"
13  Trace RAM variable "smc1.OmegaFltred"
14  Trace RAM variable "CtrlParm.qVelRef"
15  Trace RAM variable "SVGenParm.iPWMPeriod"
16  when( 1 == 0 )  Do nothing
    timeout: 999.9 secs  Stop the simulation
```

Figure 2.13: The Debug Procedures view

2.5.3 Debug Procedures view

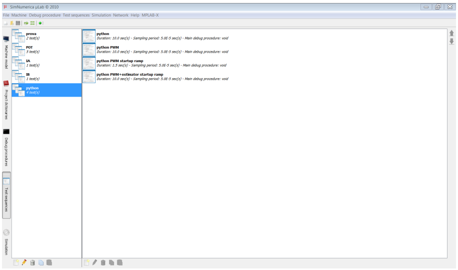
The Debug Procedures view lets you manage the debug procedures for the current test bench.

The list on the left pane displays the available debug procedures: you can click the New Debug Procedure button at the bottom to create a new one, or click an existing one to show the instructions it is composed of, and use the buttons at the bottom to rename or delete it.

The debug instructions editor works in a similar way: click the New Debug Instruction button to add a new debug instruction at the end of the debug procedures, or click an existing one and then use the buttons at the bottom of the pane to edit its contents or delete it.
More info about designing Debug Procedures can be found in Chapter 5 and in the *µLab*® Tutorial.

Please note that debug instructions are executed in a sequential order, but every new instruction is added at the bottom of the list: you can use the arrow buttons on the right to change the position of the selected debug instruction.

2.5.4 Test Sequences view



prova
2 test(s)
POT
5 test(s)
IA
1 test(s)
IB
1 test(s)
python
4 test(s)

python
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void
python PWM
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void
python PWM startup ramp
Duration: 1.5 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void
python PWM+estimator startup ramp
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

Figure 2.14: The Test Sequences view

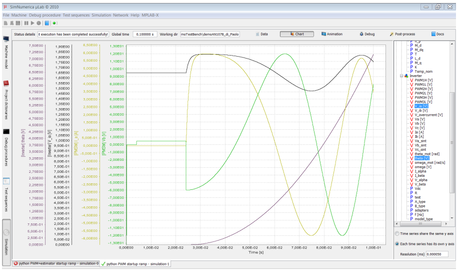
The Test Sequences view lets you manage the test sequences for the current test bench.

The list on the left pane displays the available test sequences: you can click the New Test Sequence button at the bottom to create a new one, or click an existing one to show the single tests it is composed of, and use the buttons at the bottom to rename or delete it.

The tests editor works in a similar way: click the New Test button to add a new test at the end of the sequence, or click an existing one and then use the buttons at the bottom of the pane to edit its contents or delete it.
More info about designing Test Sequences can be found in Chapter 6 and in the *µLab*® Tutorial.

2.5.5 Simulations view

The Simulations view lets you watch real-time data evolution during the execution of a test sequence.
See Chapter 7 for detailed info about simulation data.



Status details: t execution has been completed successfully!

Global time: 0.100000 s

Working dir: moTestBench\demoAN1078_di_Paolo

○ Time series share the same y axis

● Each time series has its own y axis

Resolution (ms): 0.000050

Figure 2.15: The Simulations view

2.6 The status bar

The Status Bar is shown in Figure 2.16 and contatins, from left to right, the Log Console, the Status Info, the Memory Usage icon and the name of the current *TestBench*.

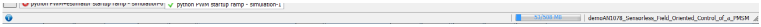
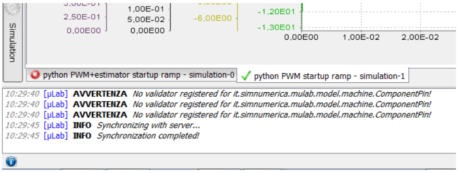

python PWM+estimator startup ramp - simulation-0
python PWM startup ramp - simulation-1
53/508 MB
demoAN1078_Sensorless_Field_Oriented_Control_of_a_PMSM

Figure 2.16: The Status bar

2.6.1 Log Console



```
10:29:40 [µLab] AVVERTENZA No validator registered for it.simnumerica.mulab.model.machine.ComponentPin!
10:29:40 [µLab] AVVERTENZA No validator registered for it.simnumerica.mulab.model.machine.ComponentPin!
10:29:40 [µLab] AVVERTENZA No validator registered for it.simnumerica.mulab.model.machine.ComponentPin!
10:29:45 [µLab] INFO Synchronizing with server...
10:29:45 [µLab] INFO Synchronization completed!
```

Figure 2.17: The Log console

Clicking the `Show Log Console` icon opens up a display area for errors, warnings and informative messages regarding the operations being performed in $\mu Lab^{®}$. It is possible to copy part of the log console contents into the system clipboard by selecting the text you want to copy, and then pressing ctrl-c or right clicking on the text display area and selecting `copy` from the context menu. If no text is selected, the entire contents of the log console are copied.

2.6.2 Status Info

Any relevant status messages are shown here.

2.6.3 Memory Usage icon

This bar shows how much memory is being used by the Java Virtual Machine.

Double clicking the Virtual Machine memory usage icon explicitly calls the Java Garbage Collector.

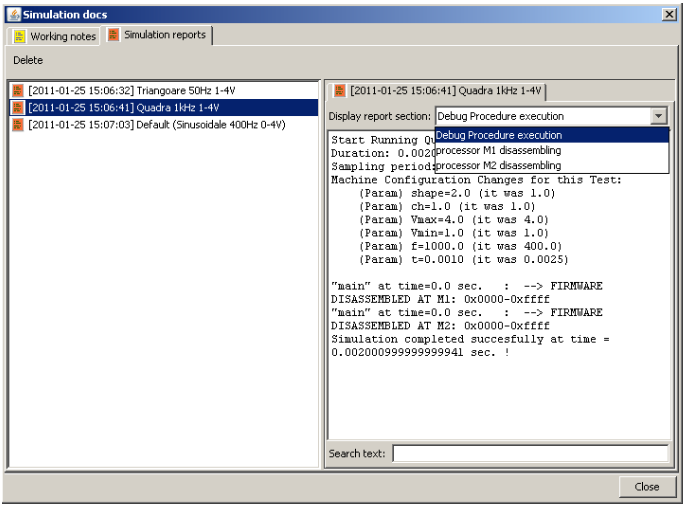
2.7 Notes and Reports

You can open the **Simulation Docs** dialog through the Simulation menu. This dialog lets you access the Notes and Reports panels.

2.7.1 Test Reports

You can specify annotations to be recorded during the execution of any debug instruction through the Debug Instruction Editor. As soon as the simulation is over, *μLab*® saves these lines in a test report: such files can be accessed and managed through the Report Manager.

See sec. 7.6 for more details about Test Reports.


Simulation docs
Working notes
Simulation reports
Delete
Display report section:
Debug Procedure execution
Debug Procedure execution
processor M1 disassembling
processor M2 disassembling
Search text:
Close

[2011-01-25 15:06:32] Triangoare 50Hz 1-4V
[2011-01-25 15:06:41] Quadra 1kHz 1-4V
[2011-01-25 15:07:03] Default (Sinusoidale 400Hz 0-4V)

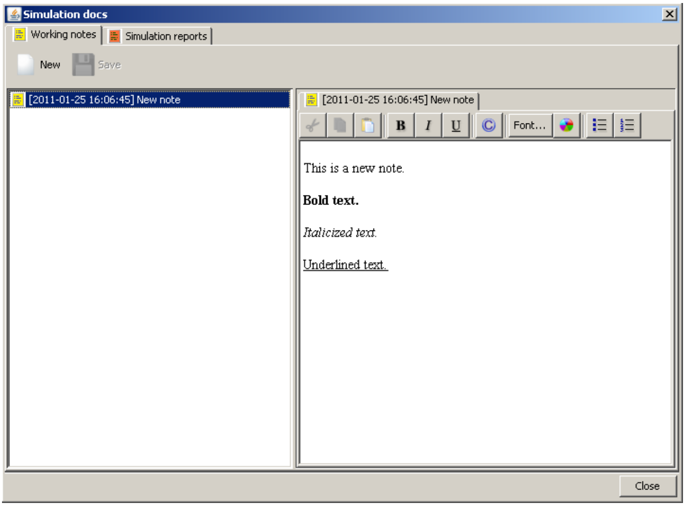
[2011-01-25 15:06:41] Quadra 1kHz 1-4V

```
Start Running Qu
Duration: 0.0020
Sampling period:
Machine Configuration Changes for this Test:
    (Param) shape=2.0 (it was 1.0)
    (Param) ch=1.0 (it was 1.0)
    (Param) Vmax=4.0 (it was 4.0)
    (Param) Vmin=1.0 (it was 1.0)
    (Param) f=1000.0 (it was 400.0)
    (Param) t=0.0010 (it was 0.0025)

"main" at time=0.0 sec.   :  --> FIRMWARE
DISASSEMBLED AT M1: 0x0000-0xffff
"main" at time=0.0 sec.   :  --> FIRMWARE
DISASSEMBLED AT M2: 0x0000-0xffff
Simulation completed succesfully at time =
0.0020009999999999941 sec. !
```

Figure 2.18: The Reports manager

2.7.2 Working Notes

μLab® features an integrated rich text editor for taking notes within the application.
Working notes belong to the test bench they are saved in: as such, when a test bench folder is moved to another *μLab*® installation, its working notes are carried over to the new copy (on the other hand, you can only access a working note from the testbench it belongs to).



This is a new note.

Bold text.

Italicized text.

Underlined text.

Figure 2.19: The Notes manager

The list on the left shows the available notes for the active test bench. Click the `New` button to create a new note, adding it to the list and opening it in the editor, or double-click an existing list entry (alternatively, select it and press the `Load` button) to open the corresponding note for editing.

See sec. 7.7 for more details about Test Reports.

Chapter 3

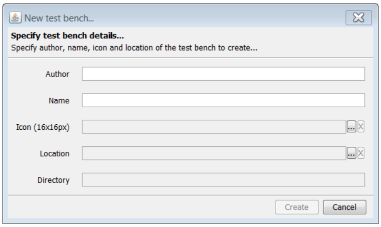
Machine/Process Modelling

3.1 Test bench

A test bench is composed of the machine model, the debug procedures and test sequences, the firmware projects and the python substitutions, the external data files, the post-processing scripts, the working notes and the test reports.
This set of objects is stored in the files inside the test bench directory: *creating (or loading) a test bench* basically means creating (or loading) such a set.

3.1.1 Creating a new test bench

To create a test bench, select `New test bench` from the `File menu`. A pop-up window appears (Figure 3.1) where you will be asked to indicate the name, the author, the image icon and the location on your hard drive of the new test bench you want to create. Once the above information is provided, you can click the Create tab and generate the test bench.



New test bench...

Specify test bench details...

Specify author, name, icon and location of the test bench to create...

Author:

Name:

Icon (16x16px):

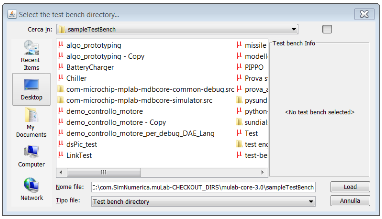
Location:

Directory:

Figure 3.1: Fill in the test bench creation form to continue

3.1.2 Loading an existing test bench

To load an existing test bench, select `Load test bench` from the `File menu`, and select the test bench you want to load through the file chooser dialog (test bench directories can be distinguished from their red “μ" icon). The selected test bench will be loaded onto the working environment.



Select the test bench directory...

Cerca in: sampleTestBench

Nome file: :\com.SimNumerica.muLab-CHECKOUT_DIRS\mulab-core-3.0\sampleTestBench

Tipo file: Test bench directory

Test bench Info

<No test bench selected>

Figure 3.2: Select a test bench and press `Load` to continue

3.2 Components

3.2.1 Component creation

In order to add a new component to the machine model, select `Add component` from the Machine menu, or press the button shown in Figure 3.3, to open the Component creation wizard (Figure 3.4). The first page of the wizard allows you to specify the name (required) and add a brief description (optional) of the new component; then press `Next`.

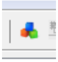


Figure 3.3: Press the `New component` button to create a new component and add it to the current machine

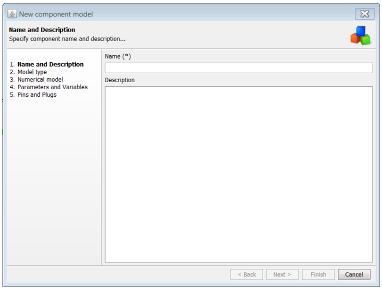
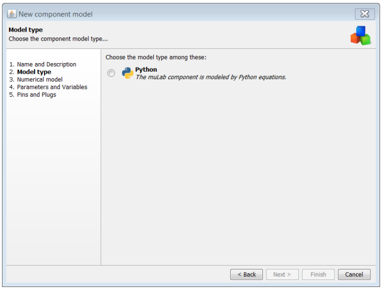


Figure 3.4: Name and description of the new component

3.2.2 Model type

Now, choose an available model type and then press `Next`. In this release of the software we consider only the *Python* model type.



New component model

Model type

Choose the component model type...

Choose the model type among these:

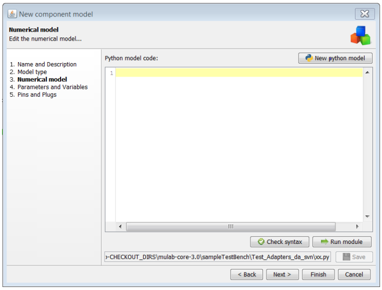
○ **Python**
The muLab component is modeled by Python equations.

Figure 3.5: Model type of the new component

3.2.3 Numerical model

The following step of the wizard requires you to formulate the numerical model of the new component.

Once you've decided the model type, you may have to specify some additional settings. This operation is strongly dependent from the model type you selected. For the *Python* model type, cfr. Figure 3.6, click the `New python model` button and choose one of the existing implementation templates, cfr. Figure 3.7.



New component model

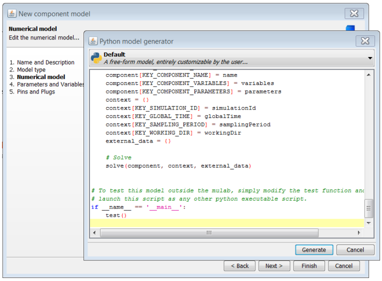
Numerical model

Edit the numerical model...

Python model code:

)-CHECKOUT_DIRS\mulab-core-3.0\sampleTestBench\Test_Adapters_da_svn\xx.py

Figure 3.6: Python component solver setup



```
    component[KEY_COMPONENT_NAME] = name
    component[KEY_COMPONENT_VARIABLES] = variables
    component[KEY_COMPONENT_PARAMETERS] = parameters
    context = {}
    context[KEY_SIMULATION_ID] = simulationId
    context[KEY_GLOBAL_TIME] = globalTime
    context[KEY_SAMPLING_PERIOD] = samplingPeriod
    context[KEY_WORKING_DIR] = workingDir
    external_data = {}

    # Solve
    solve(component, context, external_data)


# To test this model outside the mulab, simply modify the test function anc
# launch this script as any other python executable script.
if __name__ == '__main__':
    test()
```

Figure 3.7: Python component solver implementation templates chooser

3.2.3.1 Default Python model template

This template is an empty Python model with only the interface with the *μLab*® model. Write the implementation of this model between the code lines

`# <YOUR CODE (START)>: put here the actual solution of the component model.`

and

`# <YOUR CODE (END)>.`

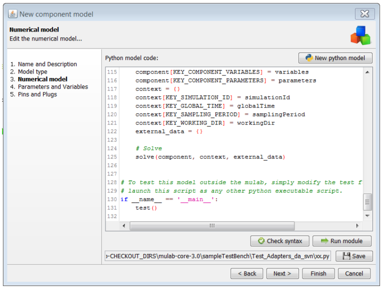
When you have done, click the `Generate` button. At this point, see Figure 3.8, the Python code is ready to be checked (by pressing the `Check syntax` button), to eventually run its test routine (by pressing the `Run module` button) if you decided to code it, and to save it in a file named `<component name>.py`.

If the code verification process, after pressing the `Check syntax` button, succeeds, you will see the pop-up of Figure 3.9; otherwise, you will see the pop-up of Figure 3.10.

Analogously, if the run of the test routine, after pressing the `Run module` button, succeeds, you will see the pop-up of Figure 3.11.

3.2.3.2 Constant Gain model template

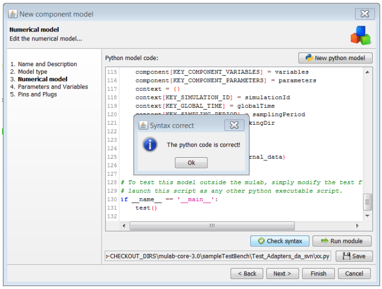
TO DO: DESCRIPTION ...



```
    component[KEY_COMPONENT_VARIABLES] = variables
    component[KEY_COMPONENT_PARAMETERS] = parameters
    context = {}
    context[KEY_SIMULATION_ID] = simulationId
    context[KEY_GLOBAL_TIME] = globalTime
    context[KEY_SAMPLING_PERIOD] = samplingPeriod
    context[KEY_WORKING_DIR] = workingDir
    external_data = {}

    # Solve
    solve(component, context, external_data)


# To test this model outside the mulab, simply modify the test f
# launch this script as any other python executable script.
if __name__ == '__main__':
    test()

```

)-CHECKOUT_DIRS\mulab-core-3.0\sampleTestBench\Test_Adapters_da_svn\xx.py

Figure 3.8: Python component solver code verification



The python code is correct!

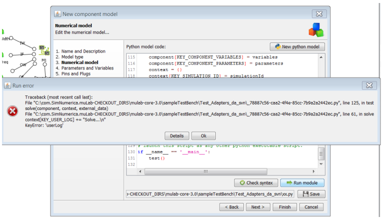
```
    component[KEY_COMPONENT_VARIABLES] = variables
    component[KEY_COMPONENT_PARAMETERS] = parameters
    context = {}
    context[KEY_SIMULATION_ID] = simulationId
    context[KEY_GLOBAL_TIME] = globalTime
                                   samplingPeriod
                                  kingDir



                                  rnal_data)


# To test this model outside the mulab, simply modify the test f
# launch this script as any other python executable script.
if __name__ == '__main__':
    test()

```

)-CHECKOUT_DIRS\mulab-core-3.0\sampleTestBench\Test_Adapters_da_svn\xx.py

Figure 3.9: Python component solver code verification success

3.2.3.3 LaTeX model template

Thanks to the *CfL Link* , *μLab*® can generate Python component models from a text written in LaTeX. Supposing you have a file latex with the component model equations written according to the *CfL*® standard, this template lets you upload the file latex from the chooser of Figure 3.12. Press the "..." button and choose your latex file, as shown in Figure 3.13 for the file `example_Lorentz.tex`, then press `Open`. Now click the `Generate` button. Wait the end of the code generation process,



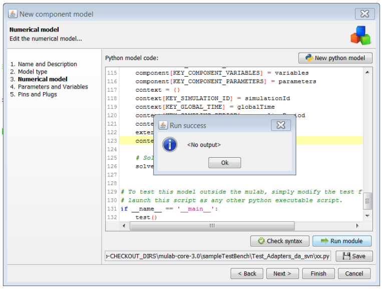
```
    component[KEY_COMPONENT_VARIABLES] = variables
    component[KEY_COMPONENT_PARAMETERS] = parameters
    context = {}
    context[KEY_SIMULATION_ID] = simulationId
```

```
Traceback (most recent call last):
File "C:\com.SimNumerica.muLab-CHECKOUT_DIRS\mulab-core-3.0\sampleTestBench\Test_Adapters_da_svn\_78887c56-caa2-4f4e-85cc-7b9a2a2442ec.py", line 125, in test
solve(component, context, external_data)
File "C:\com.SimNumerica.muLab-CHECKOUT_DIRS\mulab-core-3.0\sampleTestBench\Test_Adapters_da_svn\_78887c56-caa2-4f4e-85cc-7b9a2a2442ec.py", line 61, in solve
context[KEY_USER_LOG] += "Solve...\n"
KeyError: 'userLog'
```

```
# launch this script as any other python executable script.
if __name__ == '__main__':
    test()
```

)-CHECKOUT_DIRS\mulab-core-3.0\sampleTestBench\Test_Adapters_da_svn\xx.py

Figure 3.10: Python component solver code verification failure



```
    component[KEY_COMPONENT_VARIABLES] = variables
    component[KEY_COMPONENT_PARAMETERS] = parameters
    context = {}
    context[KEY_SIMULATION_ID] = simulationId
    context[KEY_GLOBAL_TIME] = globalTime
    conte                                    iod
    conte
    exter
    conte

    # Sol
    solve


# To test this model outside the mulab, simply modify the test f
# launch this script as any other python executable script.
if __name__ == '__main__':
    test()
```

<No output>

)-CHECKOUT_DIRS\mulab-core-3.0\sampleTestBench\Test_Adapters_da_svn\xx.py

Figure 3.11: Python component solver test routine run success

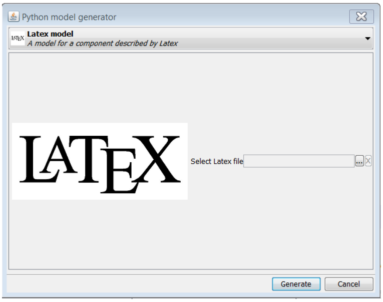
which will produce a success pop-up as in Figure 3.9.

3.2.3.4 Thermometer model template

TO DO: DESCRIPTION ...

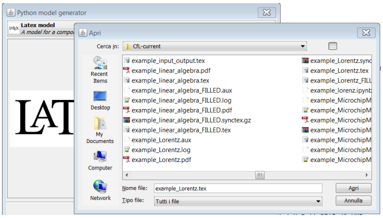
3.2.3.5 Voltage Divider model template

TO DO: DESCRIPTION ...



Latex model
A model for a component described by Latex

Figure 3.12: LaTeX component solver setup



Cerca in: CfL-current

Nome file: example_Lorentz.tex

Tipo file: Tutti i file

Figure 3.13: LaTeX component solver file chooser

3.2.3.6 Wave Generator model template

TO DO: DESCRIPTION ...

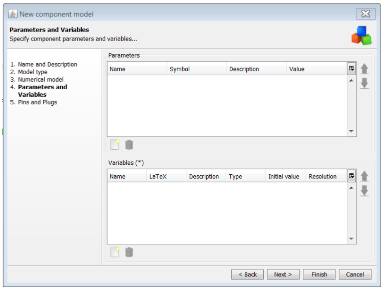
3.2.4 Parameters and variables

The third step of the wizard lets you set up parameters and variables of your new component.

Within each component, parameters and variables are uniquely identified by their name.

μLab® lets you use the same name for a parameter and a variable in the same component, or for more than one parameter (or variable) as long as each of them belongs to a different component; however, preserving name uniqueness (when possible) always remains a good practice.
You may also add a mnemonic symbol and a brief description.

To complete the setup of a parameter you have to provide its value (defaults to zero) and its unit of measurement (defaults to adimensional). Likewise, to complete the setup of a variable you have to provide its type (Input, Output or State), its initial value (defaults to zero), its unit of measurement (defaults to adimensional) and its resolution, as shown in Figure 3.14. If the component model has been created from a LaTeX file (see sec. 3.2.3.3), also its parameters and variables are automatically created and you will find them in the tables of Figure 3.14.


New component model

Parameters and Variables
Specify component parameters and variables...

1. Name and Description
2. Model type
3. Numerical model
4. **Parameters and Variables**
5. Pins and Plugs

Parameters

Name	Symbol	Description	Value

Variables (*)

Name	LaTeX	Description	Type	Initial value	Resolution

Figure 3.14: Model parameters and variables

3.2.5 Pins and plugs

The fourth step of the wizard lets you add plugs to your component.

Click on the `New` icon to create a new plug, then specify its name and pins: each pin can be associated to an I/O variable (the type of the pin

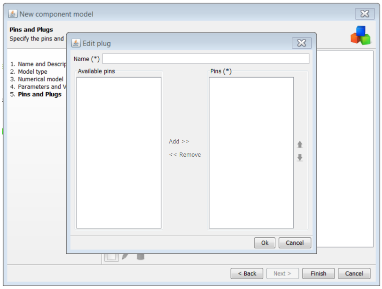


Figure 3.15: Click the `New plug` button to create a plug

is the same of the variable); you can add multiple pins to a plug, and multiple plugs to a component.

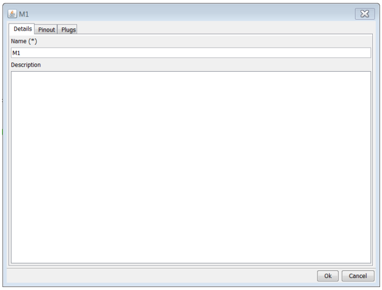
3.2.6 Post-creation editing

You may also modify the newly added component after the wizard has been completed: to open the editor, either double click the component icon on the graph, or select it and click the `Edit` item in the `Machine` menu.

3.3 Embedded Platforms

3.3.1 Embedded Platform creation

To add a new Embedded Platform, select `New embedded platform` from the Machine menu. This opens up the wizard of Figure 3.16: input the name (required) and a brief description (optional) of the new embedded platform, then press `Next` to continue.

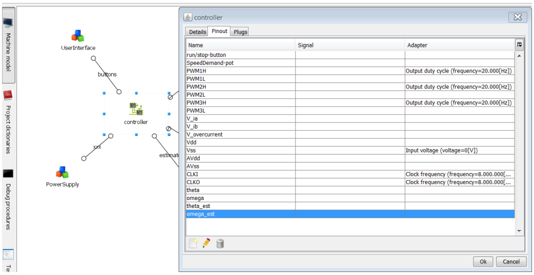


Name (*)

M1

Description

Figure 3.16: Select `New Embedded Platform` to create a new Embedded Platform and add it to the current machine



Name	Signal	Adapter
run/stop-button		
SpeedDemand-pot		
PWM1H		Output duty cycle (frequency=20.000[Hz])
PWM1L		
PWM2H		Output duty cycle (frequency=20.000[Hz])
PWM2L		
PWM3H		Output duty cycle (frequency=20.000[Hz])
PWM3L		
V_ia		
V_ib		
V_overcurrent		
Vdd		
Vss		Input voltage (voltage=0[V])
AVdd		
AVss		
CLKI		Clock frequency (frequency=8.000.000[...
CLKO		Clock frequency (frequency=8.000.000[...
theta		
omega		
theta_est		
omega_est		

Figure 3.17: Setup the pinout of the embedded platform

3.3.2 Pinout

The pinout configuration lets the user to create, modify or delete the items of the *Dictionary of processor I/O signals* (for brevity, “pin alias”) and to setup the adapters for these items.

The `Name` column displays the name of the pin alias.
The value on the `Signal` column shows the name of the model variable connected to each pin by a link (it’s empty if the pin isn’t connected to

any components).

The `Adapter` column lets you specify (and then configure) adapters for each pin alias: click on the description of the adapter (it's empty if no adapters have been configured for that pin alias) to edit it. Each adapter type is described in the following subsections.

3.3.2.1 Clock frequency

Here the user can set the external clock frequency input to the microcontroller. The adapter accepts one parameter `Clock frequency [Hz]` and no connections are required to the pin of the embedded platform model (eventual data fed up to the pin are not considered).

3.3.2.2 Input voltage

Here the user can set a constant input voltage, with no need to build an external component in the $\mu Lab^{®}$ model. The adapter accepts one parameter `Constant voltage [V]` and no connections are required to the pin of the embedded platform model (eventual data fed up to the pin are not considered).

3.3.2.3 Input/Output frequency-modulated pulse train

Here the user can generate/read a high frequency-modulated pulse train at microcontroller pins, without the need to sample the $\mu Lab^{®}$ model at the Nyquist frequency of the signal or higher. The adapter accepts one parameter `Duty cycle [%]` and reads/writes the value at the embedded platform pin as the frequency of the pulse train.

3.3.2.4 Input/Output width-modulated pulse train

Here the user can generate/read a high frequency width-modulated pulse train at microcontroller pins, without the need to sample the $\mu Lab^{®}$ model at the Nyquist frequency of the signal or higher. The adapter accepts one parameter `Frequency [Hz]` and reads/writes the value at the embedded platform pin as the duty cycle of the pulse train.

3.3.2.5 Input/Output bit stream

Here the user can generate/read a serial bit stream at microcontroller pins, without the need to sample the *μLab*® model at the Nyquist frequency of the signal or higher. The adapter accepts two parameters, `Baud rate` and `Message length` and reads/writes the values at the embedded platform pin as the bit stream.

3.3.2.6 Device pin

TO DO: ...

3.3.3 Plugs

The final step of the wizard lets you add plugs to your embedded platform.

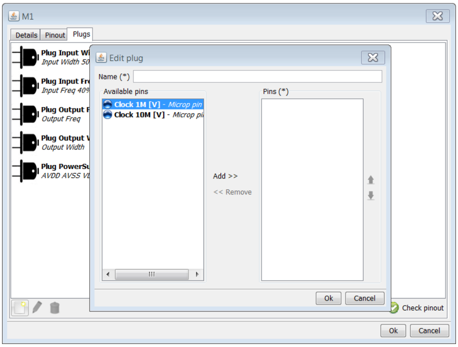
Click on the `New` icon to create a new plug, then specify its name and pins: each pin can be associated to an I/O variable (the type of the pin is the same of the variable); you can add multiple pins to a plug, and multiple plugs to an embedded platform. After all the plugs have been created, press `Ok` to end the embedded platform creation wizard.

3.3.4 Post-creation editing

You may also modify the newly added embedded platform after the wizard has been completed: to open the editor, either double click the embedded platform icon on the graph, or select it and click the `Edit` item in the `Machine` menu.

3.4 Links

In order to create a working machine, its components must be linked to each other; you can do this by creating links from the `Machine` menu (Machine → New link) or with the button shown in Figure 3.19.



M1

Plug Input Wi
Input Width 50

Plug Input Fre
Input Freq 40%

Plug Output F
Output Freq

Plug Output W
Output Width

Plug PowerSu
AVDD AVSS VI

Edit plug

Name (*)

Available pins

Clock 1M [V] - *Microp pin*

Clock 10M [V] - *Microp pi*

Pins (*)

Figure 3.18: Plug configuration dialog: specify the name of the plug and the pins it's composed of.

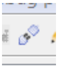


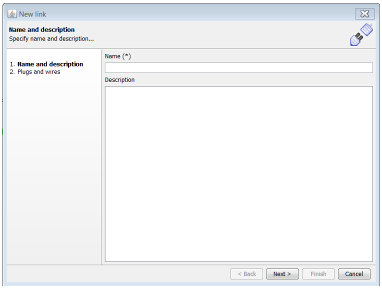
Figure 3.19: Select `New link` to start the link creation wizard

Doing so brings up the link creation wizard (Fifure 3.20). Input the name (required) and a brief description (optional) of the new link, then press `Next` to continue.

Connections between two components of any machine are formed by linking one plug on each machine with each other, as shown in Figure 3.21.

Such a connection can only be made if the two plugs are compatible. Compatibility between plugs is verified by checking if all of the pins on the first plug can be correctly connected with all of the pins on the second plug.

Two pins can be connected with each other if their I/O types are compatible (i.e. if the first is an input pin and the second is an output pin or vice versa, or if one is a microp pin) and if their units of measure-



New link

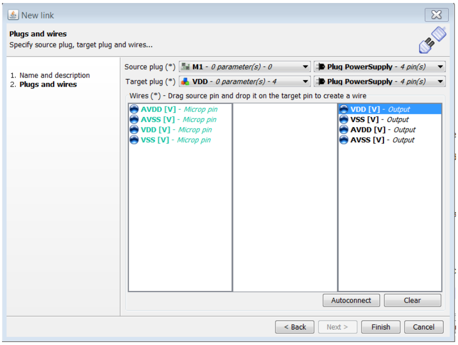
Name and description
Specify name and description...

1. **Name and description**
2. Plugs and wires

Name (*)

Description

Figure 3.20: The link creation wizard:specify the name of the new link; you may add a description text (optional). When you are done, press `Next` to continue.



New link

Plugs and wires
Specify source plug, target plug and wires...

1. Name and description
2. **Plugs and wires**

Source plug (*) **M1** - *0 parameter(s) - 0* | **Plug PowerSupply** - *4 pin(s)*

Target plug (*) **VDD** - *0 parameter(s) - 4* | **Plug PowerSupply** - *4 pin(s)*

Wires (*) - Drag source pin and drop it on the target pin to create a wire

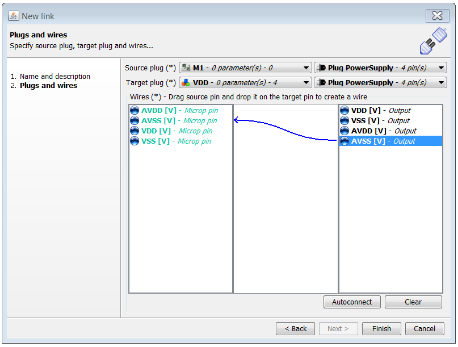
AVDD [V] - *Microp pin*
AVSS [V] - *Microp pin*
VDD [V] - *Microp pin*
VSS [V] - *Microp pin*

VDD [V] - *Output*
VSS [V] - *Output*
AVDD [V] - *Output*
AVSS [V] - *Output*

Figure 3.21: Select source and target plugs of the new link.

ment are the same. Note that the connection begins at the output pin and ends at the input pin.

Different numbers of pins, or failure to match one or more pins on one plug, will cause the two plugs to be incompatible, therefore unavailable for being connected with each other.



New link

Plugs and wires

Specify source plug, target plug and wires...

1. Name and description
2. **Plugs and wires**

Source plug (*) **M1** - *0 parameter(s) - 0* | **Plug PowerSupply** - *4 pin(s)*

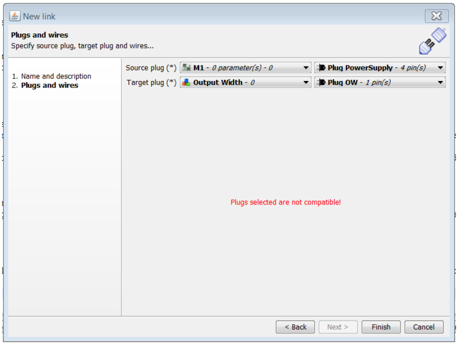
Target plug (*) **VDD** - *0 parameter(s) - 4* | **Plug PowerSupply** - *4 pin(s)*

Wires (*) - Drag source pin and drop it on the target pin to create a wire

AVDD [V] - *Microp pin*
AVSS [V] - *Microp pin*
VDD [V] - *Microp pin*
VSS [V] - *Microp pin*

VDD [V] - *Output*
VSS [V] - *Output*
AVDD [V] - *Output*
AVSS [V] - *Output*

Figure 3.22: The selected plugs are compatible: drag the output/microp pins to the input/microp pins to create wires. Press `Finish` when done.



New link

Plugs and wires

Specify source plug, target plug and wires...

1. Name and description
2. **Plugs and wires**

Source plug (*) **M1** - *0 parameter(s) - 0* | **Plug PowerSupply** - *4 pin(s)*

Target plug (*) **Output Width** - *0* | **Plug OW** - *1 pin(s)*

Plugs selected are not compatible!

Figure 3.23: The selected plugs are not compatible for connection

3.5 Groups

Selecting multiple components (by either clicking them one by one while pressing ctrl or shift, or by doing a rectangle selection) enables the group command on the toolbar

Figure 3.24: Drag a box around the components you want to group. You may now press the `Group` button to group them

Upon creating a group, you'll be asked for a group name (cannot be blank).

Figure 3.25: Specify the name of the new group and press `OK` to create it

A group can be either collapsed (only the group icon and name are shown on the graph, its contents hidden) or expanded (no group icon and name, all of its contents are shown): you can switch between the two display modes by clicking on the expand/collapse button on the top left corner. It's possible to change the contents of a group without destroying and recreating it: expand it, and drag the component you want to add (or remove, respectively) inside (or outside) the group box. If you want to remove a group, select it and press the ungroup button. Note that this only deletes the group view, not the components themselves.

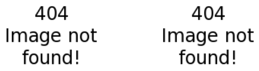


Figure 3.26: Collapsed and expanded views of a group. Press + to switch between the two modes

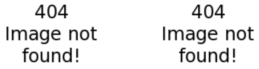


Figure 3.27: Components being added to (left picture) and removed from (right picture) a group through drag-and-drop

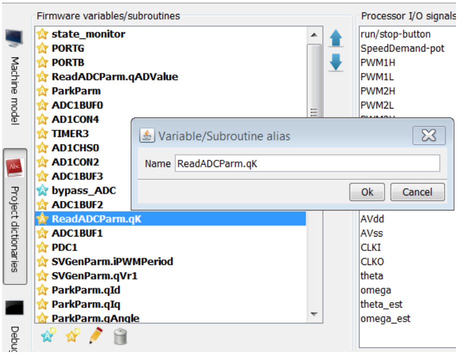
Figure 3.28: Ungrouping removes the old group relationship between objects, but the components themselves are not deleted

Chapter 4

Project Dictionaries Design

4.1 Dictionary of firmware variables/subroutines

In order to create, modify or delete an alias of a firmware variable/subroutine, use the panel and the buttons indicated in Figure 4.1.



state_monitor
PORTG
PORTB
ReadADCParm.qADValue
ParkParm
ADC1BUF0
AD1CON4
TIMER3
AD1CHS0
AD1CON2
ADC1BUF3
bypass_ADC
ADC1BUF2
ReadADCParm.qK
ADC1BUF1
PDC1
SVGenParm.iPWMPeriod
SVGenParm.qVr1
ParkParm.qId
ParkParm.qIq
ParkParm.qAngle

run/stop-button
SpeedDemand-pot
PWM1H
PWM1L
PWM2H
PWM2L
AVdd
AVss
CLKI
CLKO
theta
omega
theta_est
omega_est

Name ReadADCParm.qK

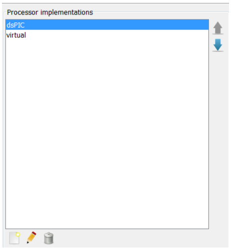
Figure 4.1: Here it is the list of firmware variable/subroutine aliases; at the left bottom there are four buttons to create a variable ("blue star"), create a subroutine ("yellow star"), modify the selected variable/subroutine ("pencil"), delete the selected variable/subroutine ("thrash"); then, press `OK` to actuate the selected action or `cancel` to abandon it.

4.2 Dictionary of processor I/O signals

In order to create, modify or delete an alias of a processor I/O signal (for brevity, "pin alias"), see sec. 3.3.2.

4.3 Dictionary of processor implementations

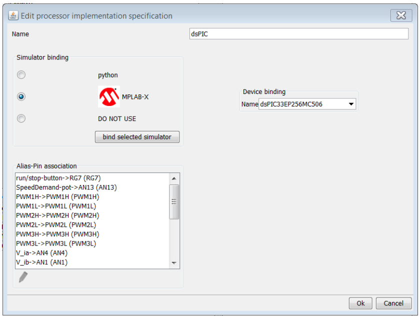
In order to create, modify or delete a processor implementation, use the panel and the buttons indicated in Figure 4.2. The editor is shown in Figure 4.3.



Processor implementations

dsPIC
virtual

Figure 4.2: Here it is the list of processor implementations existing in the project; at the left bottom there are three buttons to create, modify or delete the selected implementation.



Edit processor implementation specification

Name: dsPIC

Simulator binding

○ python
● MPLAB-X
○ DO NOT USE

Device binding

Name: dsPIC33EP256MC506

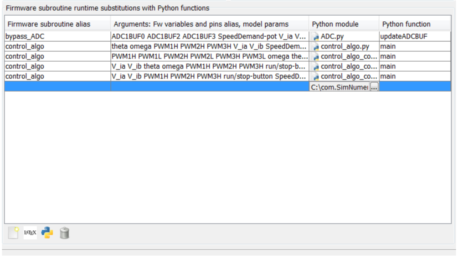
Alias-Pin association

run/stop-button->RG7 (RG7)
SpeedDemand-pot->AN13 (AN13)
PWM1H->PWM1H (PWM1H)
PWM1L->PWM1L (PWM1L)
PWM2H->PWM2H (PWM2H)
PWM2L->PWM2L (PWM2L)
PWM3H->PWM3H (PWM3H)
PWM3L->PWM3L (PWM3L)
V_ia->AN4 (AN4)
V_ib->AN1 (AN1)

Figure 4.3: The editor of a processor implementation: the user should specify a *name*, and the bindings with the *simulator*, the specific *device* inside the simulator and the alias-physical pins association.

4.4 Dictionary of runtime firmware subroutines substitutions

The dictionary of runtime firmware subroutines substitutions (see an example in Figure 4.4), is located in the left bottom of the *Project Dictionaries View* (sec. 2.5.2).



Firmware subroutine runtime substitutions with Python functions

Firmware subroutine alias	Arguments: Fw variables and pins alias, model params	Python module	Python function
bypass_ADC	ADC1BUF0 ADC1BUF2 ADC1BUF3 SpeedDemand-pot V_ia V...	ADC.py	updateADCBUF
control_algo	theta omega PWM1H PWM2H PWM3H V_ia V_ib SpeedDem...	control_algo.py	main
control_algo	PWM1H PWM1L PWM2H PWM2L PWM3H PWM3L omega the...	control_algo_co...	main
control_algo	V_ia V_ib theta omega PWM1H PWM2H PWM3H run/stop-b...	control_algo_co...	main
control_algo	V_ia V_ib PWM1H PWM2H PWM3H run/stop-button SpeedD...	control_algo_co...	main
		C:\com.SimNumer ...	

Figure 4.4: An example of a dictionary of runtime firmware subroutines substitutions

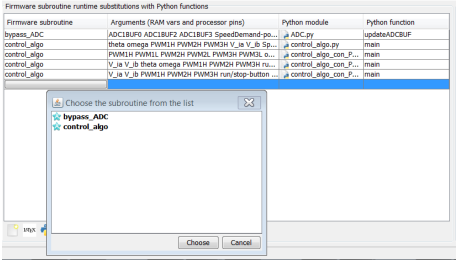
In order to:

- create a new substitution, press the leftmost button (a new item appears, as a blue table row), then click on the cells from the left to choose:
  - the alias of the firmware function to be substituted, see Fig. 4.5;
  - the arguments that could be passed to this firmware subroutine (Fig. 4.6); possible arguments are: items of the *Dictionary of firmware variables* (sec. 4.1), items of the *Dictionary of the processors I/O signals* and *model parameters* (sec. 3.2.4);
  - the python module containing the function that substitutes the firmware subroutine (4.7) and the python function itself (by writing its name in the text field).

- create a substitution from a LaTeX file, press the second button from left, and then submit the file selected through a standard file chooser;

- edit the python module, select a substitution (table row) and press the third button from the left;

- delete a substitution, press the the rightmost button (thrash icon).



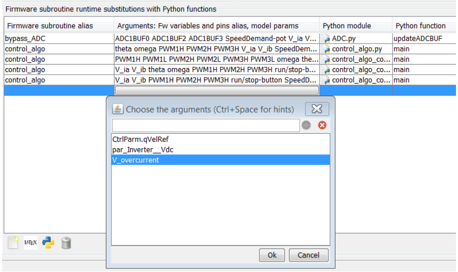
Firmware subroutine runtime substitutions with Python functions

Firmware subroutine	Arguments (RAM vars and processor pins)	Python module	Python function
bypass_ADC	ADC1BUF0 ADC1BUF2 ADC1BUF3 SpeedDemand-po...	ADC.py	updateADCBUF
control_algo	theta omega PWM1H PWM2H PWM3H V_ia V_ib Sp...	control_algo.py	main
control_algo	PWM1H PWM1L PWM2H PWM2L PWM3H PWM3L o...	control_algo_con_P...	main
control_algo	V_ia V_ib theta omega PWM1H PWM2H PWM3H ru...	control_algo_con_P...	main
control_algo	V_ia V_ib PWM1H PWM2H PWM3H run/stop-button ...	control_algo_con_P...	main

Choose the subroutine from the list

- bypass_ADC
- control_algo

Figure 4.5: Select the alias of the firmware function to be substituted, and press `Choose` to confirm the selection or `cancel` to discard it.



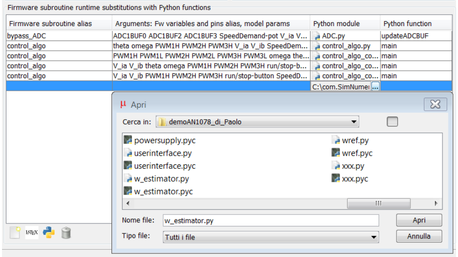
Firmware subroutine runtime substitutions with Python functions

Firmware subroutine alias	Arguments: Fw variables and pins alias, model params	Python module	Python function
bypass_ADC	ADC1BUF0 ADC1BUF2 ADC1BUF3 SpeedDemand-pot V_ia V...	ADC.py	updateADCBUF
control_algo	theta omega PWM1H PWM2H PWM3H V_ia V_ib SpeedDem...	control_algo.py	main
control_algo	PWM1H PWM1L PWM2H PWM2L PWM3H PWM3L omega the...	control_algo_co...	main
control_algo	V_ia V_ib theta omega PWM1H PWM2H PWM3H run/stop-b...	control_algo_co...	main
control_algo	V_ia V_ib PWM1H PWM2H PWM3H run/stop-button SpeedD...	control_algo_co...	main

Choose the arguments (Ctrl+Space for hints)

CtrlParm.qVelRef
par_Inverter__Vdc
V_overcurrent

Figure 4.6: Choose the arguments by typing them (`Ctrl+Space` does the auto-completion) and press the circle buttons to add or delete an argument from the list; press `OK` to confirm the selection list or `cancel` to discard it.



Firmware subroutine runtime substitutions with Python functions

Firmware subroutine alias	Arguments: Fw variables and pins alias, model params	Python module	Python function
bypass_ADC	ADC1BUF0 ADC1BUF2 ADC1BUF3 SpeedDemand-pot V_ia V...	ADC.py	updateADCBUF
control_algo	theta omega PWM1H PWM2H PWM3H V_ia V_ib SpeedDem...	control_algo.py	main
control_algo	PWM1H PWM1L PWM2H PWM2L PWM3H PWM3L omega the...	control_algo_co...	main
control_algo	V_ia V_ib theta omega PWM1H PWM2H PWM3H run/stop-b...	control_algo_co...	main
control_algo	V_ia V_ib PWM1H PWM2H PWM3H run/stop-button SpeedD...	control_algo_co...	main
		C:\com.SimNume ...	

Apri

Cerca in: demoAN1078_di_Paolo

powersupply.pyc
userinterface.py
userinterface.pyc
w_estimator.py
w_estimator.pyc
wref.py
wref.pyc
xxx.py
xxx.pyc

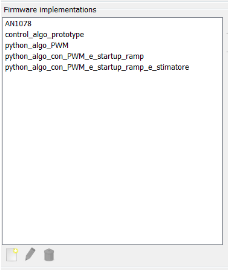
Nome file: w_estimator.py

Tipo file: Tutti i file

Figure 4.7: Select the python file with the standard chooser of the OS.

4.5 Dictionary of firmware implementations

In order to create, modify or delete a firmware implementation, use the panel and the buttons indicated in Figure 4.8. The editor of a firmware implementation is shown in Figure 4.9 and the editor of a firmware variable/subroutine binding is shown in Figure 4.10.



AN1078
control_algo_prototype
python_algo_PWM
python_algo_con_PWM_e_startup_ramp
python_algo_con_PWM_e_startup_ramp_e_stimatore

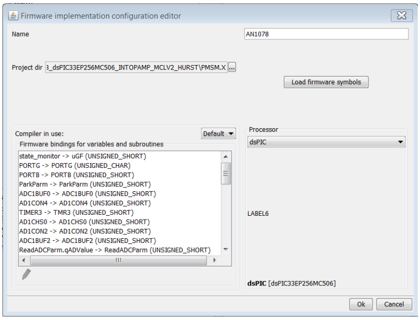
Figure 4.8: Here it is the list of firmware implementations existing in the project; at the left bottom there are three buttons to create, modify or delete the selected implementation.

4.6 Dictionary of external data files

In order to create, modify or delete a reference to an external datafile, use the panel and the buttons indicated in Figure 4.11. The editor is shown in Figure 4.14.

4.7 Dictionary of post-process scripts

In order to create, modify or delete a postprocessing script, use the panel and the buttons indicated in Figure 4.13. The editor is shown in Figure 4.14.



Firmware implementation configuration editor

Name: AN1078

Project dir: 3_dsPIC33EP256MC506_INTOPAMP_MCLV2_HURST\PMSM.X

Compiler in use: Default

Firmware bindings for variables and subroutines

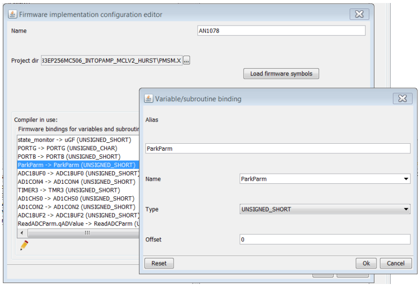
```
state_monitor -> uGF (UNSIGNED_SHORT)
PORTG -> PORTG (UNSIGNED_CHAR)
PORTB -> PORTB (UNSIGNED_SHORT)
ParkParm -> ParkParm (UNSIGNED_SHORT)
ADC1BUF0 -> ADC1BUF0 (UNSIGNED_SHORT)
AD1CON4 -> AD1CON4 (UNSIGNED_SHORT)
TIMER3 -> TMR3 (UNSIGNED_SHORT)
AD1CHS0 -> AD1CHS0 (UNSIGNED_SHORT)
AD1CON2 -> AD1CON2 (UNSIGNED_SHORT)
ADC1BUF2 -> ADC1BUF2 (UNSIGNED_SHORT)
ReadADCParm.qADValue -> ReadADCParm (UNSIGNED_SHORT)
```

Processor: dsPIC

LABEL6

dsPIC [dsPIC33EP256MC506]

Figure 4.9: The editor of a firmware implementation: the user should specify a *name*, the *project dir*, he must press the *Load firmware symbols* button, choose the compiler and the processor implementation and the bindings with the *variables* and the *subroutines* implemented in the firmware project and corresponding to the dictionary items.


Firmware implementation configuration editor
Load firmware symbols
Variable/subroutine binding
Reset
Ok
Cancel

Name AN1078

Project dir 33EP256MC506_INTOPAMP_MCLV2_HURST\PMSM.X

Compiler in use:

Firmware bindings for variables and subroutir

```
state_monitor -> uGF (UNSIGNED_SHORT)
PORTG -> PORTG (UNSIGNED_CHAR)
PORTB -> PORTB (UNSIGNED_SHORT)
ParkParm -> ParkParm (UNSIGNED_SHORT)
ADC1BUF0 -> ADC1BUF0 (UNSIGNED_SHORT)
AD1CON4 -> AD1CON4 (UNSIGNED_SHORT)
TIMER3 -> TMR3 (UNSIGNED_SHORT)
AD1CHS0 -> AD1CHS0 (UNSIGNED_SHORT)
AD1CON2 -> AD1CON2 (UNSIGNED_SHORT)
ADC1BUF2 -> ADC1BUF2 (UNSIGNED_SHORT)
ReadADCParm.qADValue -> ReadADCParm (U
```

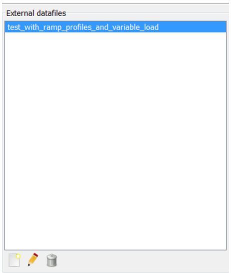
Alias ParkParm

Name ParkParm

Type UNSIGNED_SHORT

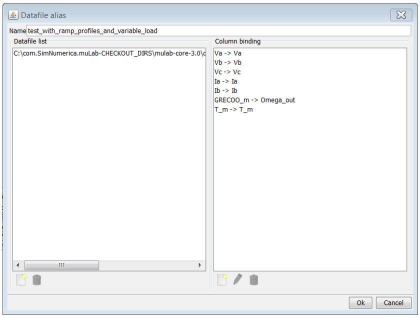
Offset 0

Figure 4.10: The editor of a firmware variable/subroutine binding: the user double click the alias of a variable/subroutine and the editor open up. He then must choose the *Name* of the variable corresponding to the selected alias, among the symbols defined in the firmware project and presented in a popup list of the editor; he must also specify the *Type* of the variable/subroutine-return-value and, if the variable is a C struct/vector, the offset **in bytes** at which the desired scalar value is allocated.



test_with_ramp_profiles_and_variable_load

Figure 4.11: Here it is the list of references to external datafiles considered in the project; at the left bottom there are three buttons to create, modify or delete the selected implementation.

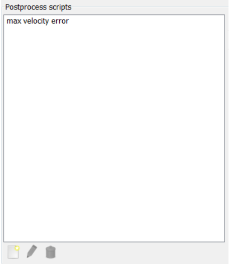


Name test_with_ramp_profiles_and_variable_load

C:\com.SimNumerica.muLab-CHECKOUT_DIRS\mulab-core-3.0\

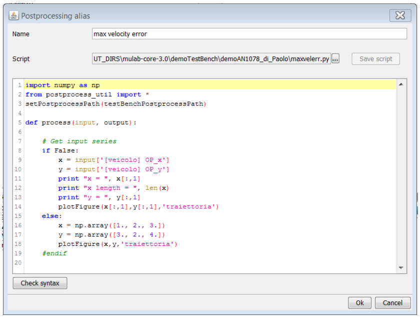
Va -> Va
Vb -> Vb
Vc -> Vc
Ia -> Ia
Ib -> Ib
GRECOO_m -> Omega_out
T_m -> T_m

Figure 4.12: The editor of a reference to an external datafile: the user should specify a *name*, the datafiles involved in this reference (usually there is a single datafile), and the bindings of each column of data with the corresponding project item, that can be: items of the *Dictionary of firmware variables* (sec. 4.1), items of the *Dictionary of the processors I/O signals* and *model parameters and variables* (sec. 3.2.4).



max velocity error

Figure 4.13: Here it is the list of the postprocessing scripts existing in the project; at the left bottom there are three buttons to create, modify or delete the selected implementation.



Name: max velocity error

Script: UT_DIRS\mulab-core-3.0\demoTestBench\demoAN1078_di_Paolo\maxvelerr.py

```
import numpy as np
from postprocess_util import *
setPostprocessPath(testBenchPostprocessPath)

def process(input, output):

    # Get input series
    if False:
        x = input['[veicolo] OP_x']
        y = input['[veicolo] OP_y']
        print "x = ", x[:,1]
        print "x length = ", len(x)
        print "y = ", y[:,1]
        plotFigure(x[:,1],y[:,1],'traiettoria')
    else:
        x = np.array([1., 2., 3.])
        y = np.array([3., 2., 4.])
        plotFigure(x,y,'traiettoria')
    #endif

```

Figure 4.14: The editor of a postprocessing script: the user should specify a *name*, the python file containing the postprocessing script, and write the python code of the script.

Chapter 5

Debug Procedures Design

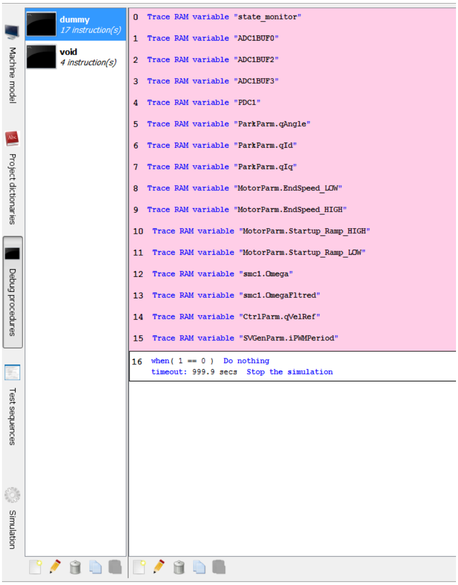
Use the navigation bar from the main $\mu Lab^{\circledR}$ dashboard to access the *Debug Procedures View* (sec. 2.5.3): it allows the user to design the Debug Procedures to be used in the project. Its structure is shown in Figure 5.1:

5.1 Creating a Debug Procedure

From the Debug Procedures page, click the `Create new debug procedure` button; then input the name of the new debug procedure and press `OK` to continue.

5.2 Renaming a Debug Procedure

To rename a debug procedure, select it from the list and click the `Edit procedure` button.



```
0  Trace RAM variable "state_monitor"
1  Trace RAM variable "ADC1BUF0"
2  Trace RAM variable "ADC1BUF2"
3  Trace RAM variable "ADC1BUF3"
4  Trace RAM variable "PDC1"
5  Trace RAM variable "ParkParm.qAngle"
6  Trace RAM variable "ParkParm.qId"
7  Trace RAM variable "ParkParm.qIq"
8  Trace RAM variable "MotorParm.EndSpeed_LOW"
9  Trace RAM variable "MotorParm.EndSpeed_HIGH"
10  Trace RAM variable "MotorParm.Startup_Ramp_HIGH"
11  Trace RAM variable "MotorParm.Startup_Ramp_LOW"
12  Trace RAM variable "smc1.Omega"
13  Trace RAM variable "smc1.OmegaFltred"
14  Trace RAM variable "CtrlParm.qVelRef"
15  Trace RAM variable "SVGenParm.iPWMPeriod"
16  when( 1 == 0 )  Do nothing
    timeout: 999.9 secs  Stop the simulation
```

Figure 5.1: The *Debug Procedures View*: there are two vertical sections, that show the list of the existing Debug Procedures (left) and the list of the Debug Instructions (right) existing in the selected Debug Procedure.

5.3 Deleting a Debug Procedure

To delete a debug procedure, select it and click the `Remove procedure` button.

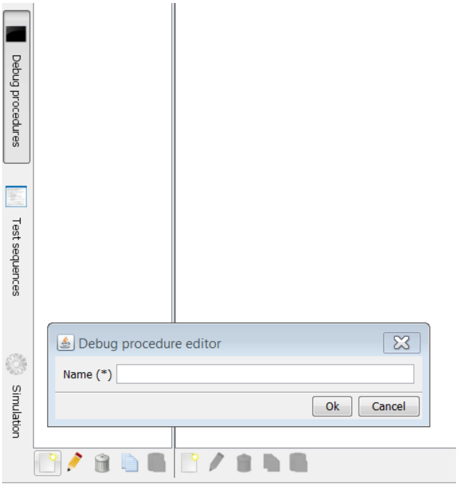
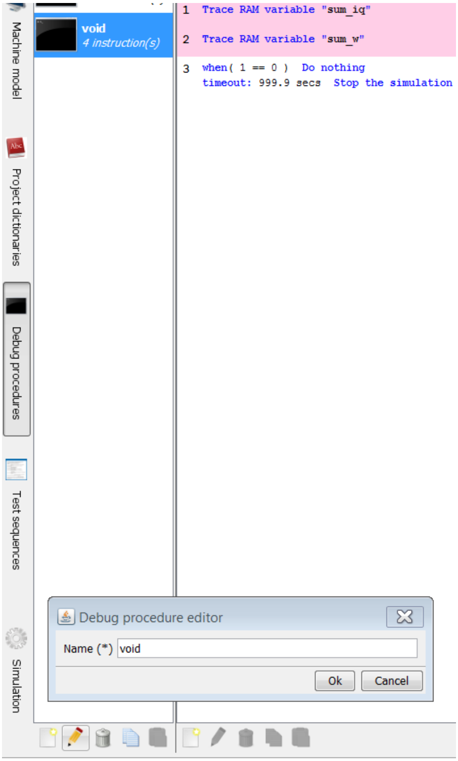


Figure 5.2: Input a name for the new Debug Procedure and press `OK` to continue

5.4 Creating/Editing/Deleting a Debug Instruction

Click the `Create new Debug Instruction` button to start the creation of a new Debug Instruction that will be added at the bottom of the instructions list of the current Debug Procedure. To edit a debug instruction, select it from the list and press the `Edit instruction` button. To remove a debug instruction from the debug procedure, select it from the list and press the `Remove instruction` button. To change the position of a Debug Instruction inside a Debug Procedure, use the arrow buttons as shown in Figure 5.6. The Debug Instruction editor is shown in Figure 5.5.

Configure the new Debug Instruction through the Debug Instruction Editor (refer to the *μLab*® tutorial for detailed information on how



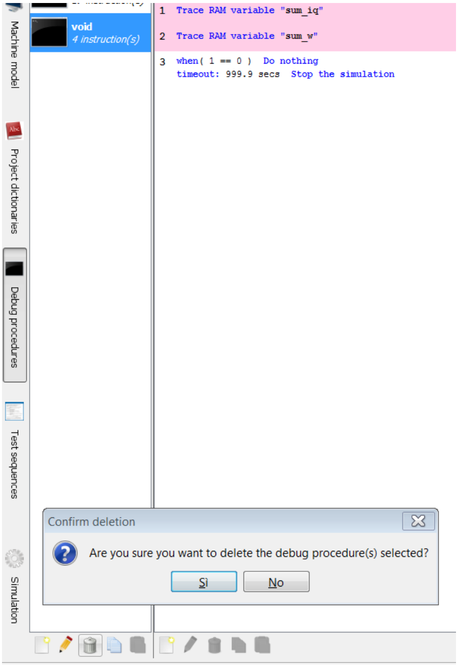
```
Trace RAM variable "sum_iq"

Trace RAM variable "sum_w"

when( 1 == 0 )  Do nothing
timeout: 999.9 secs  Stop the simulation
```

Debug procedure editor

Name (*) void

Figure 5.3: Select a Debug Procedure and click `Edit Procedure` to rename it

to configure debug instructions).



```
1  Trace RAM variable "sum_iq"

2  Trace RAM variable "sum_w"

3  when( 1 == 0 )  Do nothing
   timeout: 999.9 secs  Stop the simulation
```

Confirm deletion

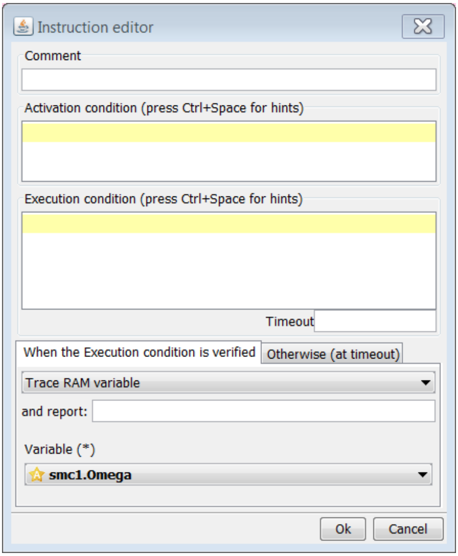
Are you sure you want to delete the debug procedure(s) selected?

Figure 5.4: Select a Debug Procedure and click `Remove Procedure` to remove it from the list

5.5 Rearranging the Debug Instruction list

You can use the `up` and `down` buttons to move up or down (by one step) a debug instruction.

Debug instruction fulfilling their activation and execution conditions will be performed in this order.



Comment

Activation condition (press Ctrl+Space for hints)

Execution condition (press Ctrl+Space for hints)

Timeout

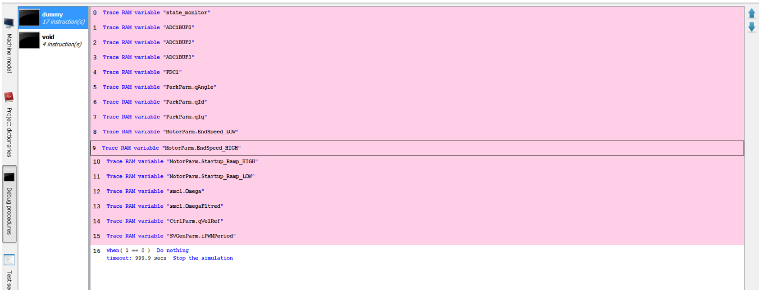
Trace RAM variable

and report:

Variable (*)

smc1.Omega

Figure 5.5: Click `Create new Debug Instruction` to add a new Debug Instruction



dummy
17 instruction(s)

void
4 instruction(s)

```
0  Trace RAM variable "state_monitor"
1  Trace RAM variable "ADC1BUF0"
2  Trace RAM variable "ADC1BUF2"
3  Trace RAM variable "ADC1BUF3"
4  Trace RAM variable "PDC1"
5  Trace RAM variable "ParkParm.qAngle"
6  Trace RAM variable "ParkParm.qId"
7  Trace RAM variable "ParkParm.qIq"
8  Trace RAM variable "MotorParm.EndSpeed_LOW"
9  Trace RAM variable "MotorParm.EndSpeed_HIGH"
10  Trace RAM variable "MotorParm.Startup_Ramp_HIGH"
11  Trace RAM variable "MotorParm.Startup_Ramp_LOW"
12  Trace RAM variable "smc1.Omega"
13  Trace RAM variable "smc1.OmegaFltred"
14  Trace RAM variable "CtrlParm.qVelRef"
15  Trace RAM variable "SVGenParm.iPWMPeriod"
16  when( 1 == 0 )  Do nothing
    timeout: 999.9 secs  Stop the simulation
```

Figure 5.6: Click the arrow buttons to move a Debug Instruction up or down the list

Chapter 6

Test Sequences Design

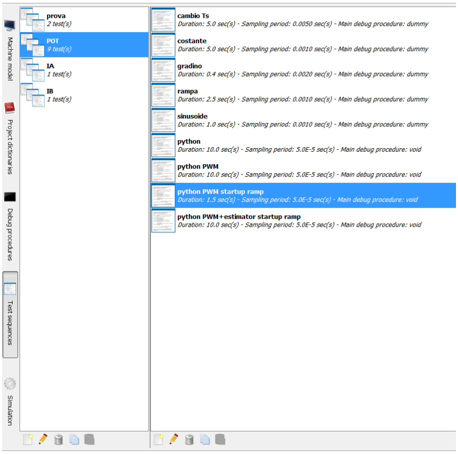
Use the navigation bar from the main *μLab*® dashboard to access the *Test Sequences View* (sec. 2.5.4): it allows the user to create and delete Test Sequences, and edit their contents. Its structure is shown in Figure 6.1:

6.1 Creating a new Test Sequence

From the Test Sequences page, click the `Create new test sequence` button; input the name of the new test sequence and press `OK` to continue.

6.2 Renaming a Test Sequence

To rename a test sequence, select it from the list and click the `Edit test sequence` button.



prova
2 test(s)

POT
9 test(s)

IA
1 test(s)

IB
1 test(s)

cambio Ts
Duration: 5.0 sec(s) - Sampling period: 0.0050 sec(s) - Main debug procedure: dummy

costante
Duration: 5.0 sec(s) - Sampling period: 0.0010 sec(s) - Main debug procedure: dummy

gradino
Duration: 0.4 sec(s) - Sampling period: 0.0020 sec(s) - Main debug procedure: dummy

rampa
Duration: 2.5 sec(s) - Sampling period: 0.0010 sec(s) - Main debug procedure: dummy

sinusoide
Duration: 1.0 sec(s) - Sampling period: 0.0010 sec(s) - Main debug procedure: dummy

python
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

python PWM
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

python PWM startup ramp
Duration: 1.5 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

python PWM+estimator startup ramp
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

Figure 6.1: The *Test Sequences View*: there are two vertical sections, that show the list of the existing Test Sequences (left) and the list of the Tests (right) defined in the selected Test Sequence.

6.3 Deleting a test sequence

To delete a test sequence, select it and click the `Remove test sequence` button.

6.4 Copy/Paste a test sequence

To copy/paste a test sequence, select it and click first the `Copy test sequence` button; then select the position to paste a copy of the Test Sequence and click the `Paste test sequence` button (Figure 6.5).

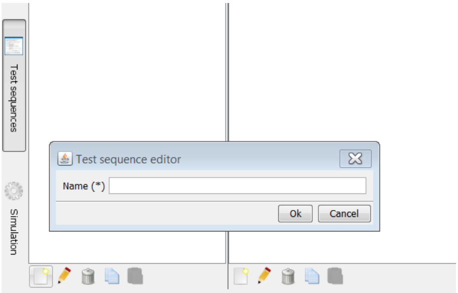


Figure 6.2: Input a name for the new Test Sequence and press `OK` to continue.

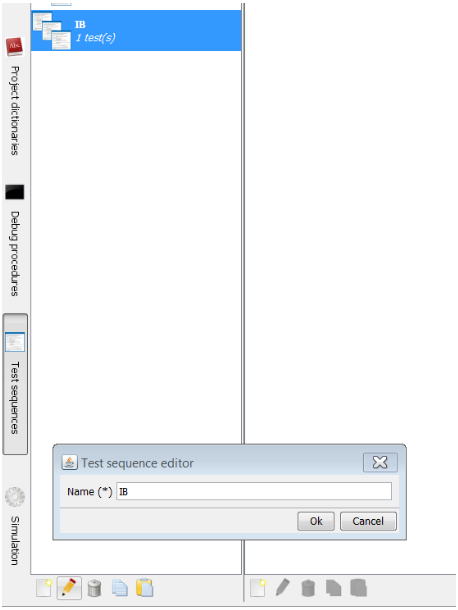
6.5 Creating/Editing/Deleting a new Test

Click the `Create new Test` button to start the creation of a new Test that will be added at the bottom of the current Test Sequence. To edit a test, select it from the list and press the `Edit test` button. To remove a test from the test sequence, select it from the list and press the `Remove test` button. The Test editor has multiple tabs:

- *Details*, , is shown in Figure 6.6;
- *Details*, , is shown in Figure 6.7;
- *Details*, , is shown in Figure 6.8;
- *Details*, , is shown in Figure 6.9;
- *Details*, , is shown in Figure 6.10;

6.6 Rearranging the Test list

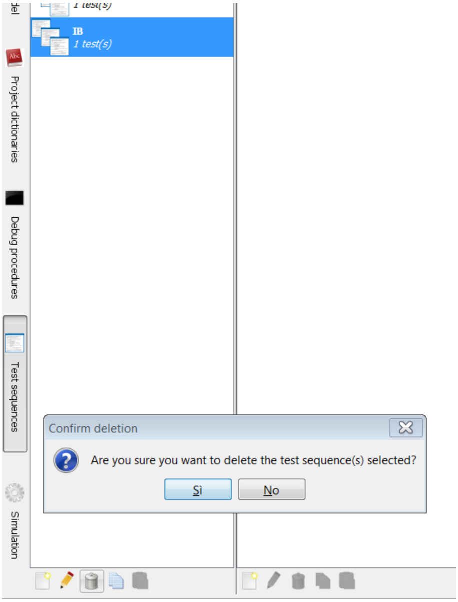
You can use the `up` and `down` buttons to move up or down (by one step) a test (Figure 6.11).



Name (*) IB

Figure 6.3: Select a Test Sequence and click `Edit test sequence` to rename it.

When a test sequence is started with the `Launch all tests` option active, tests are performed in the same order they are found in the sequence, even though they are independent (i.e. machine states don't carry over between subsequent tests).



Confirm deletion

Are you sure you want to delete the test sequence(s) selected?

Figure 6.4: Select a Test Sequence and click `Remove test sequence` to delete it.

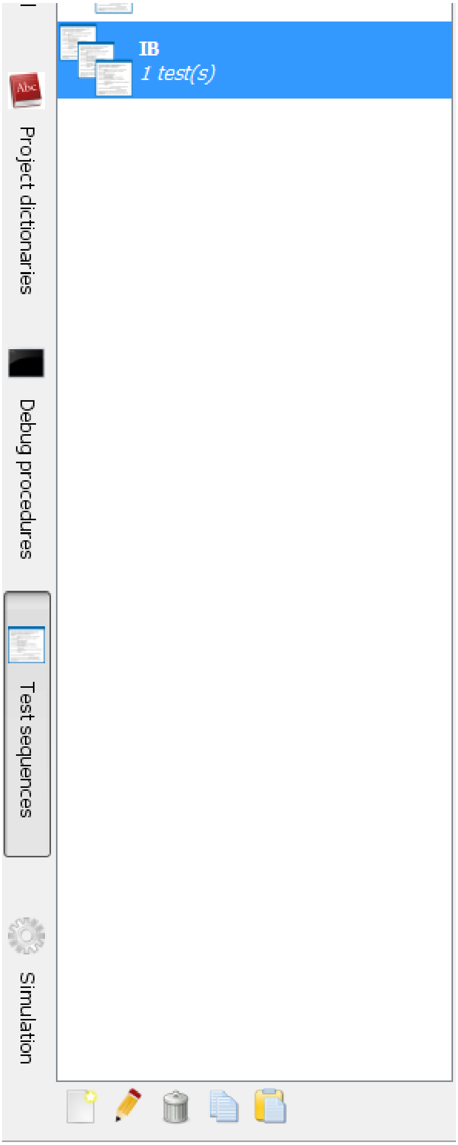
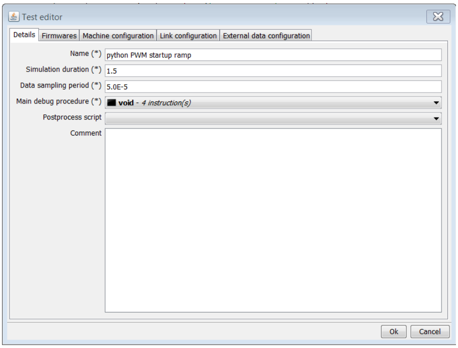
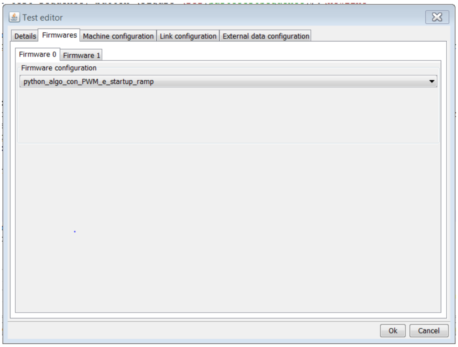


Figure 6.5: Select a Test Sequence and click `Copy test sequence` and `Paste test sequence` (rightmost buttons).



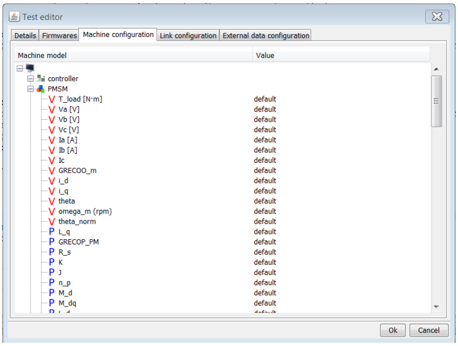
Name (*) python PWM startup ramp
Simulation duration (*) 1.5
Data sampling period (*) 5.0E-5
Main debug procedure (*) **void** - *4 instruction(s)*
Postprocess script
Comment

Figure 6.6: Click.



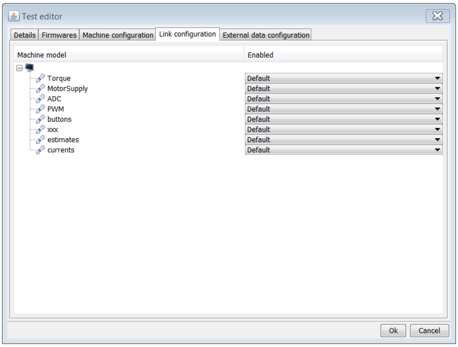
Firmware configuration
python_algo_con_PWM_e_startup_ramp

Figure 6.7: Click.



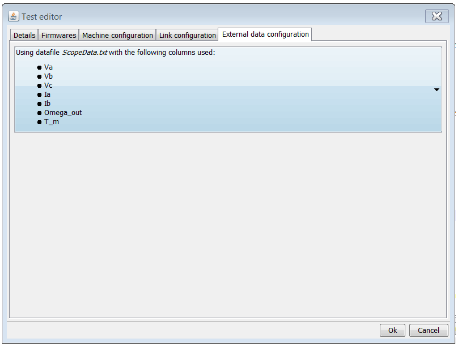
Machine model	Value
controller	
PMSM	
V T_load [N·m]	default
V Va [V]	default
V Vb [V]	default
V Vc [V]	default
V Ia [A]	default
V Ib [A]	default
V Ic	default
V GRECOO_m	default
V i_d	default
V i_q	default
V theta	default
V omega_m (rpm)	default
V theta_norm	default
P L_q	default
P GRECOP_PM	default
P R_s	default
P K	default
P J	default
P n_p	default
P M_d	default
P M_dq	default

Figure 6.8: Click `Create new Test` to add a new Test in the Test Sequence.



Machine model	Enabled
Torque	Default
MotorSupply	Default
ADC	Default
PWM	Default
buttons	Default
xxx	Default
estimates	Default
currents	Default

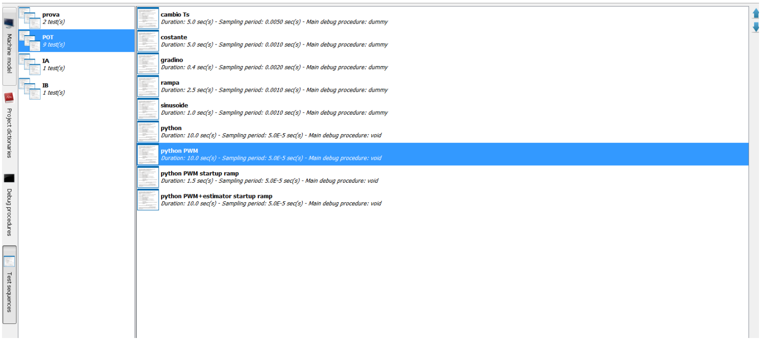
Figure 6.9: Click `Create new Test` to add a new Test in the Test Sequence.



Using datafile *ScopeData.txt* with the following columns used:

- Va
- Vb
- Vc
- Ia
- Ib
- Omega_out
- T_m

Figure 6.10: Click `Create new Test` to add a new Test in the Test Sequence.



prova
2 test(s)

POT
9 test(s)

IA
1 test(s)

IB
1 test(s)

cambio Ts
Duration: 5.0 sec(s) - Sampling period: 0.0050 sec(s) - Main debug procedure: dummy

costante
Duration: 5.0 sec(s) - Sampling period: 0.0010 sec(s) - Main debug procedure: dummy

gradino
Duration: 0.4 sec(s) - Sampling period: 0.0020 sec(s) - Main debug procedure: dummy

rampa
Duration: 2.5 sec(s) - Sampling period: 0.0010 sec(s) - Main debug procedure: dummy

sinusoide
Duration: 1.0 sec(s) - Sampling period: 0.0010 sec(s) - Main debug procedure: dummy

python
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

python PWM
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

python PWM startup ramp
Duration: 1.5 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

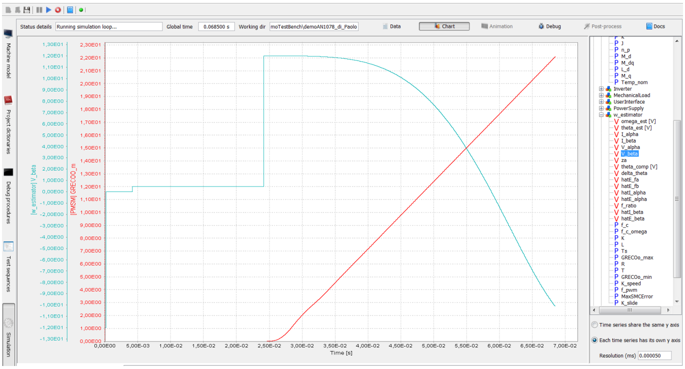
python PWM+estimator startup ramp
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

Figure 6.11: Click the arrow buttons to move a Test up or down the list

Chapter 7

Co-Simulations run

Use the navigation bar from the main *μLab*® dashboard to access the *Simulation View* (sec. 2.5.5): it allows the user to monitor the running Tests interactively, and to do some post-processing. Its structure is populated only when there is at least one Test running or finished, and it is shown in Figure 7.1:


Status details: Running simulation loop...

Global time: 0.068500 s

Working dir: moTestBench\demoAN1078_di_Paolo

○ Time series share the same y axis

● Each time series has its own y axis

Resolution (ms): 0.000050

Figure 7.1: The *Test Simulation View* :

7.1 Starting Simulation

In order to launch a Simulation, you have to perform a few basic steps, described in the following subsections.

7.1.1 Test selection

There are two ways to select a Test or a Test Sequence:

- From any view in μ*Lab*® : click the *Launch Test* or the *Launch Test Sequence* buttons (Figure 7.2) to show the Test Sequence selection screen (Figure 7.3). Note that pressing the *Launch Test Sequence* button causes all of the contained Tests to be run one after another, sequentially. Moreover, every test is unrelated from the others (i.e. machine states don't carry over between two consecutive tests).

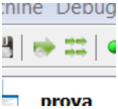
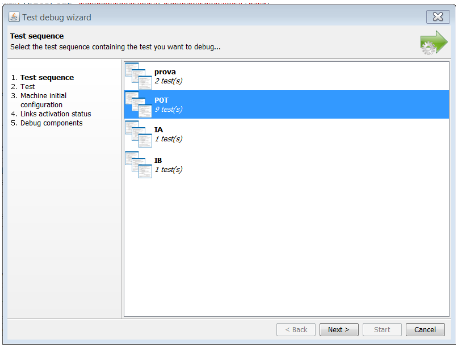


Figure 7.2: The *Launch Test* (left) and the *Launch Test Sequence* (right) buttons.

- From the *Test Sequences View*: if a Test Sequence and eventually also a Test are selected in this view, click the *Launch Test* or the *Launch Test Sequence* buttons; the selection made will bypass the Test Sequence selection screen (Figure 7.3) and bypass also the Test selection screen (Figure 7.4), if also a Test is selected in this view or the *Launch Test Sequence* button has been pressed.



Test sequence
Select the test sequence containing the test you want to debug...

1. **Test sequence**
2. Test
3. Machine initial configuration
4. Links activation status
5. Debug components

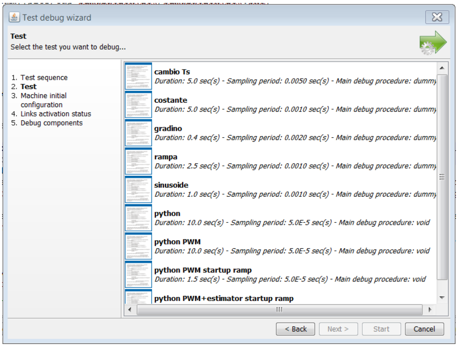
prova
2 test(s)

POT
9 test(s)

IA
1 test(s)

IB
1 test(s)

Figure 7.3: Choose a test sequence from the list and click `Next`



Test
Select the test you want to debug...

1. Test sequence
2. **Test**
3. Machine initial configuration
4. Links activation status
5. Debug components

cambio Ts
Duration: 5.0 sec(s) - Sampling period: 0.0050 sec(s) - Main debug procedure: dummy

costante
Duration: 5.0 sec(s) - Sampling period: 0.0010 sec(s) - Main debug procedure: dummy

gradino
Duration: 0.4 sec(s) - Sampling period: 0.0020 sec(s) - Main debug procedure: dummy

rampa
Duration: 2.5 sec(s) - Sampling period: 0.0010 sec(s) - Main debug procedure: dummy

sinusoide
Duration: 1.0 sec(s) - Sampling period: 0.0010 sec(s) - Main debug procedure: dummy

python
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

python PWM
Duration: 10.0 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

python PWM startup ramp
Duration: 1.5 sec(s) - Sampling period: 5.0E-5 sec(s) - Main debug procedure: void

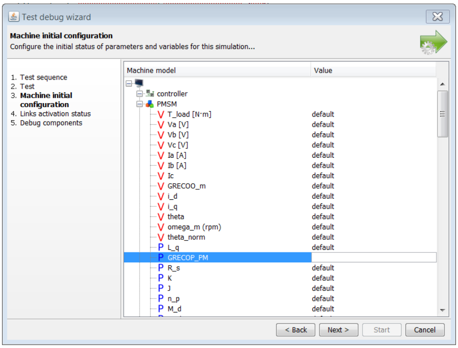
python PWM+estimator startup ramp

Figure 7.4: Select the test you want to execute and click `Next`

7.1.2 Machine configuration

Machine variables and parameters are displayed in a tree-structured table, grouped by component (and by processor model in case they belong to an embedded platform); any values specified here will override both the starting default values (only for the execution of the selected

Test or Test Sequence) and the values in the specific Test configurations.



Machine initial configuration

Configure the initial status of parameters and variables for this simulation...

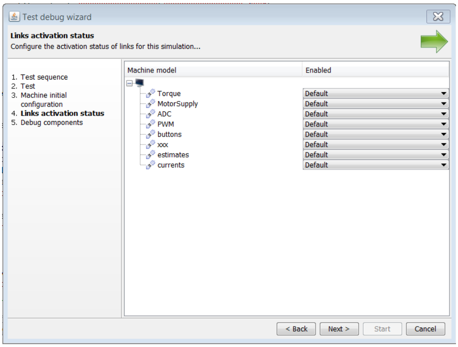
1. Test sequence
2. Test
3. **Machine initial configuration**
4. Links activation status
5. Debug components

Machine model	Value
controller	
PMSM	
V T_load [N·m]	default
V Va [V]	default
V Vb [V]	default
V Vc [V]	default
V Ia [A]	default
V Ib [A]	default
V Ic	default
V GRECOO_m	default
V i_d	default
V i_q	default
V theta	default
V omega_m (rpm)	default
V theta_norm	default
P L_q	default
P GRECOP_PM	
P R_s	default
P K	default
P J	default
P n_p	default
P M_d	default

Figure 7.5: Configure the starting values of variables and parameters

7.1.3 Links configuration

Each link between model components can be here enabled or disabled; any values specified here will override both the starting default values (only for the execution of the selected Test or Test Sequence) and the values in the specific Test configurations.



Links activation status

Configure the activation status of links for this simulation...

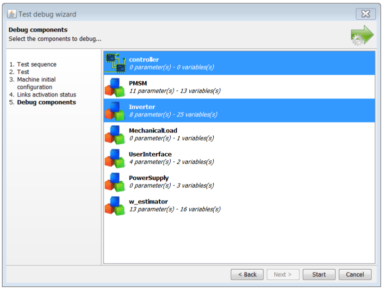
1. Test sequence
2. Test
3. Machine initial configuration
4. **Links activation status**
5. Debug components

Machine model	Enabled
Torque	Default
MotorSupply	Default
ADC	Default
PWM	Default
buttons	Default
xxx	Default
estimates	Default
currents	Default

Figure 7.6: Configure the starting values of variables and parameters

7.1.4 Debug components

Select which components you want to debug during the execution of the simulation (select multiple components by pressing CTRL while clicking them). Note that only the components that support debugging will be available for selection.



Test debug wizard

Debug components
Select the components to debug...

1. Test sequence
2. Test
3. Machine initial configuration
4. Links activation status
5. **Debug components**

controller
0 parameter(s) - 0 variables(s)

PMSM
11 parameter(s) - 13 variables(s)

Inverter
8 parameter(s) - 25 variables(s)

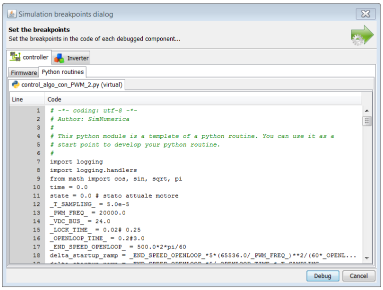
MechanicalLoad
0 parameter(s) - 1 variables(s)

UserInterface
4 parameter(s) - 2 variables(s)

PowerSupply
0 parameter(s) - 3 variables(s)

w_estimator
13 parameter(s) - 16 variables(s)

Figure 7.7: Select the components you want to debug by clicking them while pressing CTRL.



Simulation breakpoints dialog

Set the breakpoints
Set the breakpoints in the code of each debugged component...

```
# -*- coding: utf-8 -*-
# Author: SimNumerica
#
# This python module is a template of a python routine. You can use it as a
# start point to develop your python routine.
#
import logging
import logging.handlers
from math import cos, sin, sqrt, pi
time = 0.0
state = 0.0 # stato attuale motore
_T_SAMPLING_ = 5.0e-5
_PWM_FREQ_ = 20000.0
_VDC_BUS_ = 24.0
_LOCK_TIME_ = 0.02# 0.25
_OPENLOOP_TIME_ = 0.2#3.0
_END_SPEED_OPENLOOP_ = 500.0*2*pi/60
delta_startup_ramp = _END_SPEED_OPENLOOP_*5*(65536.0/_PWM_FREQ_)**2/(60*_OPENL...
```

Figure 7.8: Select the components you want to debug by clicking them while pressing CTRL.

7.2 Simulation flow

The simulation flow can be controlled with the buttons shown in Figure 7.2.

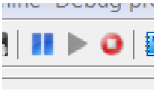


Figure 7.9: The flow control buttons, from left to right: `Pause`, `Resume` and `Cancel`

7.2.1 Pausing Simulations

Use this command to pause the execution of the Test shown in the active tab.

7.2.2 Resuming Simulations

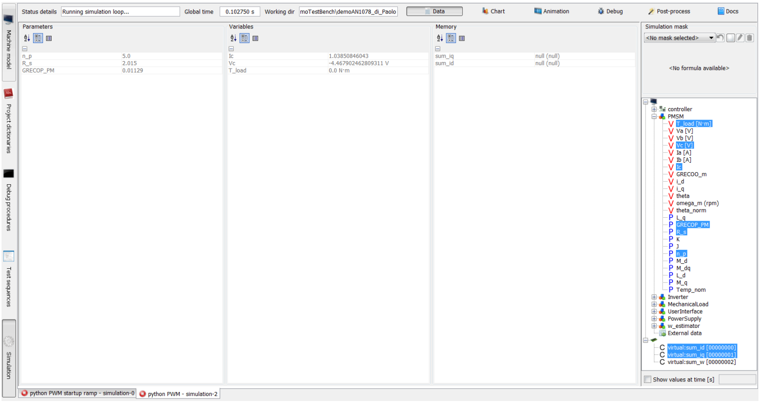
Use this command to resume the execution of the Test shown in the active tab.
This action can only be performed if the Test was previously paused: an aborted Test cannot be resumed.

7.2.3 Aborting Simulations

Use this command to stop and abort the execution of the current Test.
Aborting a Test also cancels the creation of its Test Reports.

7.3 Data visualization

7.3.1 Data view



Status details: Running simulation loop...

Global time: 0.102750 s

Working dir: moTestBench\demoAN1078_di_Paolo

Parameters

n_p	5.0
R_s	2.015
GRECOP_PM	0.01129

Variables

Ic	1.03850846043
Vc	-4.467902462809311 V
T_load	0.0 N·m

Memory

sum_iq	null (null)
sum_id	null (null)

Simulation mask: <No mask selected>

<No formula available>

☐ Show values at time [s]

Figure 7.10: The data view is composed of two main sections: on the left, the data tables; on the right, the selection panes

Select from the menu on the right the data you want to watch by clicking them; hold CTRL while clicking to select multiple items.

The Data view shows the last value taken by the selected items; specifying an instant in the text field on the right lets you see the values taken by the variables at that time.
This feature is only available in the data view, and can only be used to show values at one given instant, not during an interval.

The Data view also supports Simulation Masks: analogue to Machine Masks, they can be used to filter the values shown in the tables, which is especially useful when dealing with complex models described by large quantities of data.

7.3.2 Chart view

The chart view gives you a real-time graphical representation of the model variables and parameters, updated instant by instant while the *Test* is being executed. The timeline starts at the beginning of the simulation,

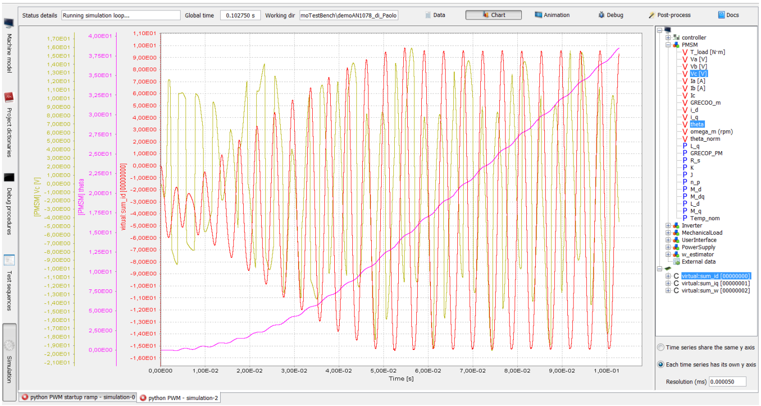
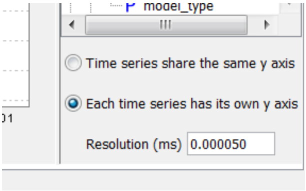

Status details: Running simulation loop...
Global time: 0.102750 s
Working dir: moTestBench\demoAN1078_di_Paolo
○ Time series share the same y axis
● Each time series has its own y axis
Resolution (ms): 0.000050

Figure 7.11: The Chart view

and goes all the way to the current instant: you can zoom on a specific region by indicating it with the mouse pointer, and scrolling with the wheel (scroll up to zoom in, scroll down to zoom out).
Note that the minimum zoom level is the original one; click on the graph to automatically reset the zoom.

You can select the variables you want to be shown in a graph by clicking on their names in the selection panel: press ctrl while clicking for multiple selection.

Two display modes for data series are supported: one with a common y-axis, when viewing comparable data, and one with separate y-axes, when the data you're viewing cannot be compared (i.e. because of different units of measurement or different orders of magnitude). If you want to save or print a graph, right-click the graph area and select `Save`



○ Time series share the same y axis

● Each time series has its own y axis

Resolution (ms) 0.000050

Figure 7.12: Common and separate y-axis modes buttons

or `Print` from the context menu.

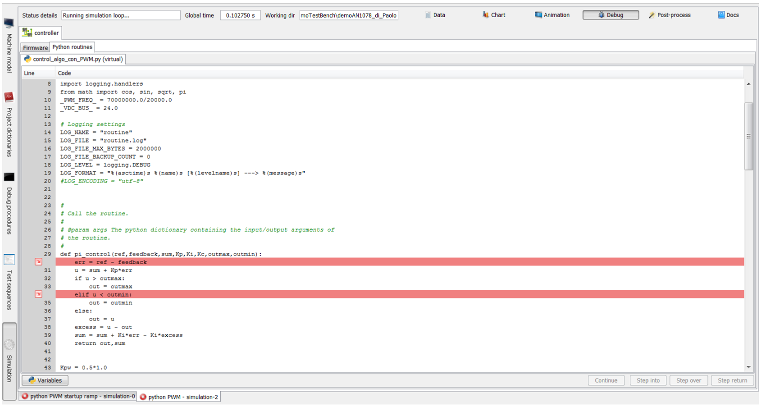
7.3.3 Animation view

This functionality is forthcoming.

This view lets you use the simulation data as input variables for a graphical animation. Custom animations can be added to the built-in set: write the Java FX script that implements your animation, opportunely binding its variables to the simulation parameters, and place the script file in the test bench folder.

7.3.4 Debug view

The debug view lets you modify breakpoints and watch variables, parameters and memories while the simulation is running. Click on a line number/instruction address to add a breakpoint on that line; click again to remove it. When the execution of the platform firmware or component solver stops at a breakpoint (Figure 7.14), you can use the control buttons to either skip that breakpoint, do one step (that is, execute the current instruction and stop at the next one), or continue the simulation (until the following breakpoint is reached), see Figure 7.15.

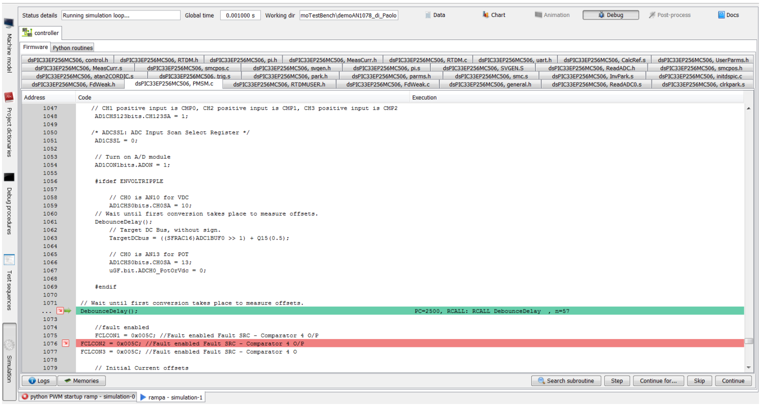


Status details: Running simulation loop...
Global time: 0.102750 s
Working dir: moTestBench\demoAN1078_di_Paolo

```
import logging.handlers
from math import cos, sin, sqrt, pi
_PWM_FREQ_ = 70000000.0/20000.0
_VDC_BUS_ = 24.0

# Logging settings
LOG_NAME = "routine"
LOG_FILE = "routine.log"
LOG_FILE_MAX_BYTES = 2000000
LOG_FILE_BACKUP_COUNT = 0
LOG_LEVEL = logging.DEBUG
LOG_FORMAT = "%(asctime)s %(name)s [%(levelname)s] ---> %(message)s"
#LOG_ENCODING = "utf-8"


#
# Call the routine.
#
# @param args The python dictionary containing the input/output arguments of
# the routine.
#
def pi_control(ref,feedback,sum,Kp,Ki,Kc,outmax,outmin):
    err = ref - feedback
    u = sum + Kp*err
    if u > outmax:
        out = outmax
    elif u < outmin:
        out = outmin
    else:
        out = u
    excess = u - out
    sum = sum + Ki*err - Ki*excess
    return out,sum


Kpw = 0.5*1.0
```

Figure 7.13: The Debug view

Click the `Memories` button to show a watch table for variables data (Figure 7.16).


Status details
Running simulation loop...
Global time
0.001000 s
Working dir
moTestBench\demoAN1078_di_Paolo
Data
Chart
Animation
Debug
Post-process
Docs
Machine model
Project dictionaries
Debug procedures
Test sequences
Simulation
controller
Firmware
Python routines
dsPIC33EP256MC506, control.h
dsPIC33EP256MC506, RTDM.h
dsPIC33EP256MC506, pi.h
dsPIC33EP256MC506, MeasCurr.h
dsPIC33EP256MC506, RTDM.c
dsPIC33EP256MC506, uart.h
dsPIC33EP256MC506, CalcRef.s
dsPIC33EP256MC506, UserParms.h
dsPIC33EP256MC506, MeasCurr.s
dsPIC33EP256MC506, smcpos.c
dsPIC33EP256MC506, svgen.h
dsPIC33EP256MC506, pi.s
dsPIC33EP256MC506, SVGEN.S
dsPIC33EP256MC506, ReadADC.h
dsPIC33EP256MC506, smcpos.h
dsPIC33EP256MC506, atan2CORDIC.s
dsPIC33EP256MC506. trig.s
dsPIC33EP256MC506, park.h
dsPIC33EP256MC506, parms.h
dsPIC33EP256MC506, smc.s
dsPIC33EP256MC506, InvPark.s
dsPIC33EP256MC506, initdspic.c
dsPIC33EP256MC506, FdWeak.h
dsPIC33EP256MC506, PMSM.c
dsPIC33EP256MC506, RTDMUSER.h
dsPIC33EP256MC506, FdWeak.c
dsPIC33EP256MC506, general.h
dsPIC33EP256MC506, ReadADC0.s
dsPIC33EP256MC506, clrkpark.s
Address
Code
Execution
Logs
Memories
Search subroutine
Step
Continue for...
Skip
Continue
python PWM startup ramp - simulation-0
rampa - simulation-1

```
    // CH1 positive input is CMP0, CH2 positive input is CMP1, CH3 positive input is CMP2
     AD1CHS123bits.CH123SA = 1;

    /* ADCSSL: ADC Input Scan Select Register */
     AD1CSSL = 0;

     // Turn on A/D module
     AD1CON1bits.ADON = 1;

     #ifdef ENVOLTRIPPLE

         // CH0 is AN10 for VDC
         AD1CHS0bits.CH0SA = 10;
     // Wait until first conversion takes place to measure offsets.
     DebounceDelay();
         // Target DC Bus, without sign.
         TargetDCbus = ((SFRAC16)ADC1BUF0 >> 1) + Q15(0.5);

         // CH0 is AN13 for POT
         AD1CHS0bits.CH0SA = 13;
         uGF.bit.ADCH0_PotOrVdc = 0;

     #endif

// Wait until first conversion takes place to measure offsets.
DebounceDelay();                                                        PC=2500, RCALL: RCALL DebounceDelay  , n=57

     //fault enabled
     FCLCON1 = 0x005C; //Fault enabled Fault SRC - Comparator 4 O/P
FCLCON2 = 0x005C; //Fault enabled Fault SRC - Comparator 4 O/P
FCLCON3 = 0x005C; //Fault enabled Fault SRC - Comparator 4 O

     // Initial Current offsets
```

Figure 7.14: Active breakpoints in the embedded firmware co-simulation

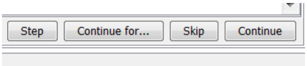
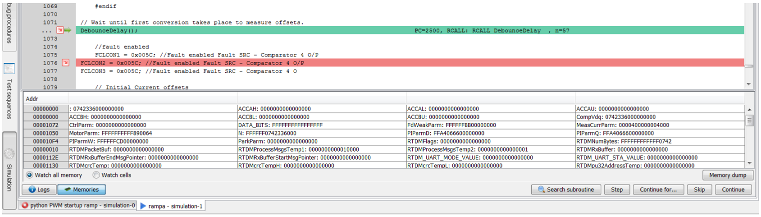


Figure 7.15: Debug control buttons



```
     #endif

// Wait until first conversion takes place to measure offsets.
DebounceDelay();                                                        PC=2500, RCALL: RCALL DebounceDelay  , n=57

     //fault enabled
     FCLCON1 = 0x005C; //Fault enabled Fault SRC - Comparator 4 O/P
FCLCON2 = 0x005C; //Fault enabled Fault SRC - Comparator 4 O/P
FCLCON3 = 0x005C; //Fault enabled Fault SRC - Comparator 4 O

     // Initial Current offsets
```

Addr				
00000000	: 0742336000000000	ACCAH: 0000000000000000	ACCAL: 0000000000000000	ACCAU: 0000000000000000
00000000	ACCBH: 0000000000000000	ACCBL: 0000000000000000	ACCBU: 0000000000000000	CompVdq: 0742336000000000
00001072	CtrlParm: 0000000000000000	DATA_BITS: FFFFFFFFFFFFFFFF	FdWeakParm: FFFFFFBB00000000	MeasCurrParm: 0000400000004000
00001050	MotorParm: FFFFFFFFFF890064	N: FFFFFF0742336000	PIParmD: FFA4066600000000	PIParmQ: FFA4066600000000
000010F4	PIParmW: FFFFFFCD00000000	ParkParm: 0000000000000000	RTDMFlags: 0000000000000000	RTDMNumBytes: FFFFFFFFFFFF0742
00000010	RTDMPacketBuf: 0000000000000000	RTDMProcessMsgsTemp1: 0000000000010000	RTDMProcessMsgsTemp2: 0000000000000001	RTDMRxBuffer: 0000000000000000
0000112E	RTDMRxBufferEndMsgPointer: 0000000000000000	RTDMRxBufferStartMsgPointer: 0000000000000000	RTDM_UART_MODE_VALUE: 0000000000000000	RTDM_UART_STA_VALUE: 0000000000000000
00001130	RTDMcrcTemp: 0000000000000000	RTDMcrcTempH: 0000000000000000	RTDMcrcTempL: 0000000000000000	RTDMpu32AddressTemp: 0000000000000000

● Watch all memory ○ Watch cells

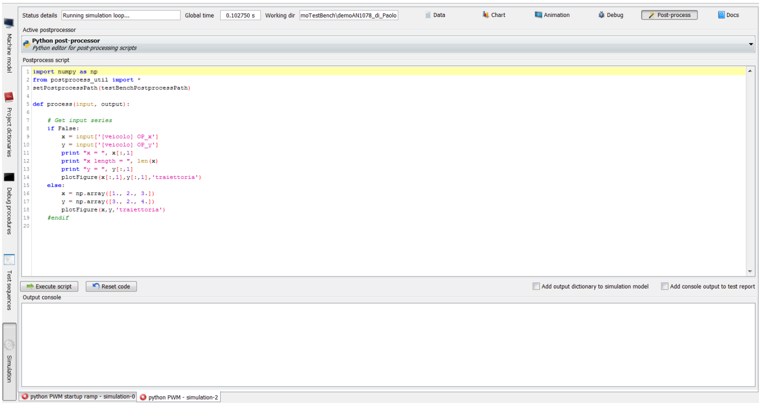
Figure 7.16: Click the `Memories` button to watch variables

7.4 Co-simulation logs

TO DO: ...

7.5 Post Processing

TO DO: ...



Status details: Running simulation loop...

Global time: 0.102750 s

Working dir: moTestBench\demoAN1078_di_Paolo

Active postprocessor: **Python post-processor** *Python editor for post-processing scripts*

Postprocess script

```
import numpy as np
from postprocess_util import *
setPostprocessPath(testBenchPostprocessPath)

def process(input, output):

    # Get input series
    if False:
        x = input['[veicolo] OP_x']
        y = input['[veicolo] OP_y']
        print "x = ", x[:,1]
        print "x length = ", len(x)
        print "y = ", y[:,1]
        plotFigure(x[:,1],y[:,1],'traiettoria')
    else:
        x = np.array([1., 2., 3.])
        y = np.array([3., 2., 4.])
        plotFigure(x,y,'traiettoria')
    #endif

```

□ Add output dictionary to simulation model

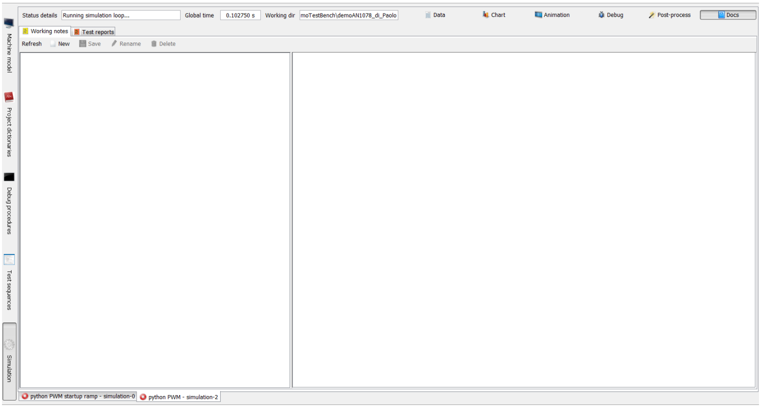
□ Add console output to test report

Output console

Figure 7.17: The Post Processing simulation view:

7.6 Test Reports

Test Reports are automatically created during the execution of Simulations.
Open the `Simulation Docs` window and click the `Test Reports` tab to access the Reports Manager.



Status details: Running simulation loop...
Global time: 0.102750 s
Working dir: moTestBench\demoAN1078_di_Paolo

Figure 7.18: Docs simulation view: the `Test Reports` tab

7.6.1 Reports Manager

The layout of the Reports Manager is composed of two parts:

- on the left, the list of the Test Reports stored in the active testbench;
- on the right, the Test Report editor.

Reports can be opened in the editor by double-clicking their name on the list;

groups instead can be expanded or collapsed by double-clicking them.

Right-clicking the tab of a Report in the editor shows a pop-up menu you can use to close that report, close all of the other reports, or close all of the open reports.

Select a Report on the list and click the `Delete` button to delete the selected Report.

Click the `Search Reports` button to open the advanced search tools window.

7.6.2 Text search tools

The *μLab*® Reports Manager provides you with two text search tools.

You may search for a string in the currently active Report by using the `Search text` field at the bottom of the Report panel.
Any text matching the specified string will be then highlighted in the active Report, and the editor will scroll to show the first match instance.

If you need to search for a string in all of the Reports stored in the active testbench, without opening every single one of them in the editor, open the advanced search tools window by clicking the `Search Reports` button.
Specify the string you want to search in the `Search text` field, select one of the two result display modes, and click `Ok` to perform the search:

- `search reports and show list` looks for matches with the specified string and displays a list of the Reports containing any matches; you may then pick which Reports to open by selecting them on the list (press CTRL while clicking them for multiple selections) and clicking `Choose`;
- `search reports and open them` looks for matches with the specified string and automatically open all of the Reports containing

any matches.

7.6.3 Report Groups management

Report Groups are created by default when you execute a whole Test Sequence; a Report generated by the execution of a single Test (that is, by using the `Launch just 1 test in the sequence` setting), by default, does not belong to any Report Groups.

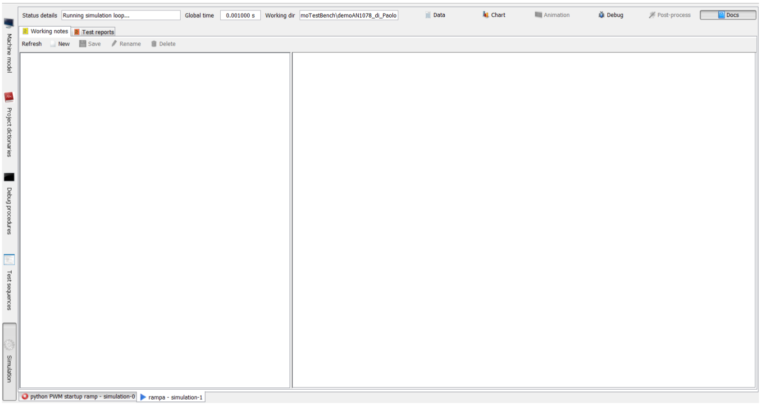
Double click on a Group to expand or collapse it: expanding a Group shows a list of the Reports it contains.

Select a set of Reports (press CTRL while clicking the single Reports for multiple selection) and click the `Create` button to create a new group containing the selected Reports;
select an existing group and click `Rename` to rename it;
select an existing group and click `Dissolve` to remove the Group only: the Reports it contained will become single Reports;
select an existing group and click `Delete` to remove the Group and delete the Reports it contains.

7.7 Working Notes

Working Notes are automatically created during the execution of Simulations.

Open the `Simulation Docs` window and click the `Working Notes` tab.



Status details: Running simulation loop...

Global time: 0.001000 s

Working dir: moTestBench\demoAN1078_di_Paolo

Figure 7.19: Docs simulation view: the `Working Notes` tab